(4)

CONTRACT REPORT BRL-CR-605

BRL

1938 – Serving the Army for Fifty Years – 1988

DYNAMICS OF A BALLOTING PROJECTILE IN A MOVING GUN TUBE

K. A. ANSARI
J. W. BAUGH, JR.

DECEMBER 1988

U.S. ARMY LABORATORY COMMAND

BALLISTIC RESEARCH LABORATORY

ABERDEEN PROVING GROUND, MARYLAND

DESTRUCTION NOTICE

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) BRL-CR-605	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Battelle Pacific NW Lab 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) P.O. Box 999 Richland, WA 99352	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION MSB, IBD, BRL 8b. OFFICE SYMBOL (If applicable) SLCBR-IB-M	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Director U.S. Army Ballistic Research Lab Aberdeen Proving Ground, MD 21005-5066	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 1L162618A PROJECT NO. 1L1626 18AH80 TASK NO. WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Dynamics of a Balloting Projectile in a Moving Gun Tube

12. PERSONAL AUTHOR(S)
K.A. Ansari and J.W. Baugh, Jr.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
CR	FROM _____ TO _____	January 1986	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

In recent years, there have been significant strides made in modeling gun dynamics. Current capabilities include the use of flexible gun tube and projectile descriptions. However, modeling the interface conditions between the tube and projectile is still an area of very active research. Highly detailed finite element models, which are currently under development at the BRL and at other agencies will lead to a much better understanding of the important parameters involved in the balloting of projectiles. These models, however, require both substantial computer resources and time. Thus there is a need for the development of simplified tube/projectile interface descriptions which both realistically represent the interface and are easily implemented. The bourrelet/gun-bore and obturator models incorporated in this study reasonably satisfy these requirements.

continued on next page

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Edward M. Patton	22b TELEPHONE (Include Area Code) (301) 278-6805
	22c. OFFICE SYMBOL SLCBR-IB-M

DD Form 1473, JUN 86 Previous editions are obsolete

19. **ABSTRACT** - cont'd

A nonlinear dynamic analysis of a balloting projectile is presented in this report. The mathematical model used is a six-degree-of-freedom coupled model of the projectile and gun tube system. The effects of obturator flexibility and projectile impact with the gun bore at the bourrelet are included in the analysis, and the nonlinear differential equations of motion of the system are derived using a Lagrangian formulation. Gaussian elimination and Newmark's constant-average-acceleration method are employed to obtain a solution. Finally, numerical results for some specific test cases are obtained and discussed.

Keywords: computer simulation; computer models; interior ballistics; (KI)

DYNAMICS OF A BALLOTING PROJECTILE IN A MOVING GUN TUBE

by

KA Ansari
JW Baugh, Jr.

Pacific Northwest Lab
Richland, Washington

Prepared for

U.S. Army Ballistic Research Laboratory
Aberdeen Proving Grounds
Aberdeen, Maryland

January 1986

TABLE OF CONTENTS

PAGE

LIST OF ILLUSTRATIONS.....5

I. INTRODUCTION.....7

II. LITERATURE REVIEW.....9

III. SYSTEM DESCRIPTION.....10

IV. THE SYSTEM KINETIC ENERGY.....10

V. PROJECTILE/GUN-BORE CONTACT MODEL.....13

VI. OBTURATOR MODEL.....14

VII. THE SYSTEM POTENTIAL ENERGY.....15

VIII. EQUATIONS OF MOTION.....15

IX. SOLUTION TECHNIQUE.....19

X. COMPUTER CODE DESCRIPTION.....21

XI. NUMERICAL RESULTS AND DISCUSSION.....21

XII. CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER STUDY.....32

ACKNOWLEDGEMENT.....34

REFERENCES.....35

APPENDIX A.....37

APPENDIX B.....41

APPENDIX C.....45

APPENDIX D.....51

LIST OF SYMBOLS.....77

DISTRIBUTION LIST.....79

LIST OF ILLUSTRATIONS

Figure

1 Displaced Position of Gun Bore. 11

2 Projectile with Bourrelet and Obturator within Gun Bore 11

3 Body-fixed Axes and Projectile Displacements. 12

4 Obturator and Bourrelet/Gun-Bore Contact Models 13

5 Time History of Gas Pressure on Projectile. 23

6 Projectile Foundation Moment. 23

7 Time History of Horizontal Gun Displacement (X_g). 24

8 Time History of Vertical Gun Displacement (Y_g). 24

9 Time History of X-Displacement of Projectile (x_p) 25

10 Time History of Gun Angular Displacement (θ) for Case 1a. 26

11 Time History of Gun Angular Displacement (θ) for Case 1b. 26

12 Time History of Gun Angular Displacement (θ) for Case 2a. 27

13 Time History of Gun Angular Displacement (θ) for Case 2b. 27

14 Time History of Y-Displacement of Projectile (y_p) for Case 1a . . . 28

15 Time History of Y-Displacement of Projectile (y_p) for Case 1b . . . 28

16 Time History of Y-Displacement of Projectile (y_p) for Case 2a . . . 29

17 Time History of Y-Displacement of Projectile (y_p) for Case 2b . . . 29

18 Time History of Yawing Motion of Projectile (α) for Case 1a. . . . 30

19 Time History of Yawing Motion of Projectile (α) for Case 1b 30

20 Time History of Yawing Motion of Projectile (α) for Case 2a 31

21 Time History of Yawing Motion of Projectile (α) for Case 2b 31

22 Bourrelet Impacting Bore. 44

I. INTRODUCTION

The yawing or wobbling motion of a projectile within a gun tube is an important consideration in internal ballistics and is known as balloting. This motion is a function of a number of small, difficult-to-measure parameters such as manufacturing tolerances, lack of concentricity of the engraving of the obturator, projectile and tube deformation, obturation of the propellant gases and obturator wear. Certain combinations of these parameters can potentially cause malfunctions to occur which include poor projectile launch, damage to the fuse mechanism and explosion in the bore. The yawing motion of the projectile leads to yaw at shot exit, and the resultant inaccuracy of the round. It can also lead to muzzle wear which can be significant for large bore diameters and muzzle velocities. As a wobbling projectile moves down a gun barrel, several forces act on it influencing its motion. These can be due to imbalances, asymmetries, clearances and deformation in the projectile itself, engraving and frictional forces, propellant gas blowby, tube vibration, recoil, and coupling between the projectile and the tube motion. Depending upon the magnitudes involved, these can lead to excessive local tube wear, excessive projectile engraving, malfunctioning of projectile components, and undesirable in-bore motion. They can also cause improper exterior ballistic free flight of the projectile. Differences occur between the point of aim and the point of impact, which can be ascribed to the motion of the gun itself, the aerodynamic forces acting and the yawing motion of the projectile in the gun bore. Although these differences may be only as large as a few mils, they must, obviously, be minimized. In order for this to be effected, a realistic prediction of the yawing motion of the projectile in the gun tube becomes necessary.

II. LITERATURE SURVEY

In order to predict the first maximum yaw exterior to the gun, Reno[1] carries out a rigorous Lagrangian treatment of the angular motion of the projectile in the bore. He assumes that the projectile is centered at the rotating band, the plane of yaw rotates with the rifling, the impact of the bourrelet is normal to the tube and rebound is described by a coefficient of restitution. Friction between the projectile and the bore is ignored. The projectile is approximated by a single degree of freedom pendulum system, and a closed-form solution is obtained. However, the initial yaw observed during development of a 36-inch mortar in 1944 was greater than that predicted by Reno's theory. Furthermore, scratches on the bore surface indicated that the projectile precessed in a direction opposite to the spin imparted by the rifling. In an attempt to provide an explanation, Thomas[2] generalizes Reno's approach by removing the constraint on the orientation of yaw and deduces the motion of the plane of yaw. His results show little difference in the yaw; the orientation of the plane of yaw, however, is quite different. Darpas[3] studies a situation in which the bourrelet touches the bore throughout the projectile travel within the gun tube and concludes that the gyroscopic effect due to the projectile being cocked in the bore, is a predominant one. The eccentricity of the center of gravity and the droop of the tube may also not be negligible. Perdreaville[4] presents an analysis of the projectile balloting problem using Lagrange's equations and Euler angles. The projectile is assumed to have no center of mass offset and to be in dynamic balance about the principal geometric axes. The equations are written to represent complete

---

[1] *F.V. Reno, "The Motion of the Axis of a Spinning Shell Inside the Bore of a Gun," BRL Report No. BRL-R-320, 1943 (AD# 491839).*

[2] *L.E. Thomas, "The Motion of the Axis of a Spinning Shell Inside the Bore of a Gun," BRL Report No. BRL-R-544, 1945 (PB# 22102).*

[3] *J.G. Darpas, Translated by H.P. Hitchcook, "Transverse Forces on Projectiles Which Rotate in the Barrel," BRL Report No. BRL-MR-1208, 1959 (Memorial de l'artillerie francaise, 31:19, No. 1, 1957) (AD# 218873).*

[4] *F.J. Perdreaville, "Analysis of the Lateral Motion of a Projectile in the Gun Tube," Sandia Laboratories, Albuquerque, NM, SC-RR-710071, 1971.*

lateral dynamic freedom of the projectile bourrelet within the bore. Impact is included and the equations are set-up so that the motion of the gun tube can also be introduced. Chu and Soechting[5] present the same theory although they resort to Euler's dynamical equations. Perdreaville[6] extends his analysis of Reference 4 to include the effects due to an unbalanced projectile in a rigid gun tube. In a subsequent document[7] he develops equations of motion describing the lateral motion of an artillery projectile as it moves down an elastic gun tube that vibrates laterally. However, he does not generate any numerical results in either of these reports. Langhaar and Boresi[8] analyze the problem of dynamics of a projectile in a concentric flexible moving tube. The motion of the tube is accounted for by using the Kirchhoff-Clebsch theory of deformed rods. However, the effect of projectile balloting is ignored and no numerical results are shown.

This report presents a dynamic analysis of a balloting projectile in a moving gun tube. A six degree of freedom mathematical model for the projectile/gun-tube system is resorted to and the nonlinear differential equations of motion derived using a Lagrangian approach. The effects of obturator flexibility, and projectile impact with the bore at the bourrelet or at the obturator along with subsequent rebound are included. The nonlinear equations of motion are solved using Gaussian elimination and the constant-average-acceleration integration scheme. Finally, numerical results for some sample test cases are obtained and discussed.

---

[5] *S.H. Chu and F.K. Soechting, "Transverse Motion of an Accelerating Shell," Picatinny Arsenal, Dover, NJ, PA-TR-4314, 1972 (AD# 894572).*

[6] *F.J. Perdreaville, "Analysis of the Lateral Motion of an Unbalanced Projectile in a Rigid Gun Tube," Sandia Laboratories, Albuquerque, NM, 87115, SAND 74-0361, 1974.*

[7] *F.J. Perdreaville, "Analysis of the Lateral Motion of an Unbalanced Projectile in an Elastic Gun Tube," Sandia Laboratories, Albuquerque, NM, 87115, SAND 74-0362, 1974.*

[8] *H.L. Langhaar and A.P. Boresi, "Dynamics of a Projectile in a Concentric Flexible Tube," BLM Applied Mechanics Consultants, Contract Report No. ARBRL-CR-00501, February 1983.*

III. SYSTEM DESCRIPTION

The mathematical model employed is a six-degree-of-freedom coupled model of the gun-tube and projectile system. An inertial reference frame XYZ is selected with the gun-tube center of mass at (X_g, Y_g). The bore may be displaced from its original position due to recoil and vibration with the displaced position described by (X_g, Y_g, θ) as shown in Figure 1. A typical projectile is represented in Figure 2. The origin of the body-fixed axes (x,y,z) is fixed on the bore centerline as shown in Figure 3. The motion of the projectile is described by the coordinates (x_p, y_p) of its center of mass relative to the gun and the yawing angle α about the orthogonal axis z. The projectile is assumed to have no center of mass offset and to be in dynamic balance about its principal axes. Because of recoil and vibration, the gun bore can have axial as well as lateral and rotational motion. As the projectile moves down the gun bore, the bourrelet as well as the obturator can impact with the gun bore and then rebound from it. This effect can be included by using simple restoring springs to represent the interaction between the bore and the projectile. Both the gun tube and projectile are assumed to behave as rigid bodies and rifling of the projectile is not considered.

IV. THE SYSTEM KINETIC ENERGY

The total kinetic energy of the projectile/gun-tube system is

$$T = T_{gun} + T_{projectile}$$

$$= \tfrac{1}{2} m_g (\dot{X}_g^2 + \dot{Y}_g^2) + \tfrac{1}{2} I_g \dot{\theta}^2$$

$$+ \tfrac{1}{2} m_p (\dot{X}_p^2 + \dot{Y}_p^2) + \tfrac{1}{2} I_p \dot{\alpha}^2 \qquad (1)$$

where m_g, m_p represent the masses of the gun tube and projectile and I_g, I_p represent the mass moments of inertia about axes orthogonal to the plane of motion at the respective centers of gravity. Expressions for the quantities $\dot{X}_p$ and $\dot{Y}_p$ are derived in Appendix A.

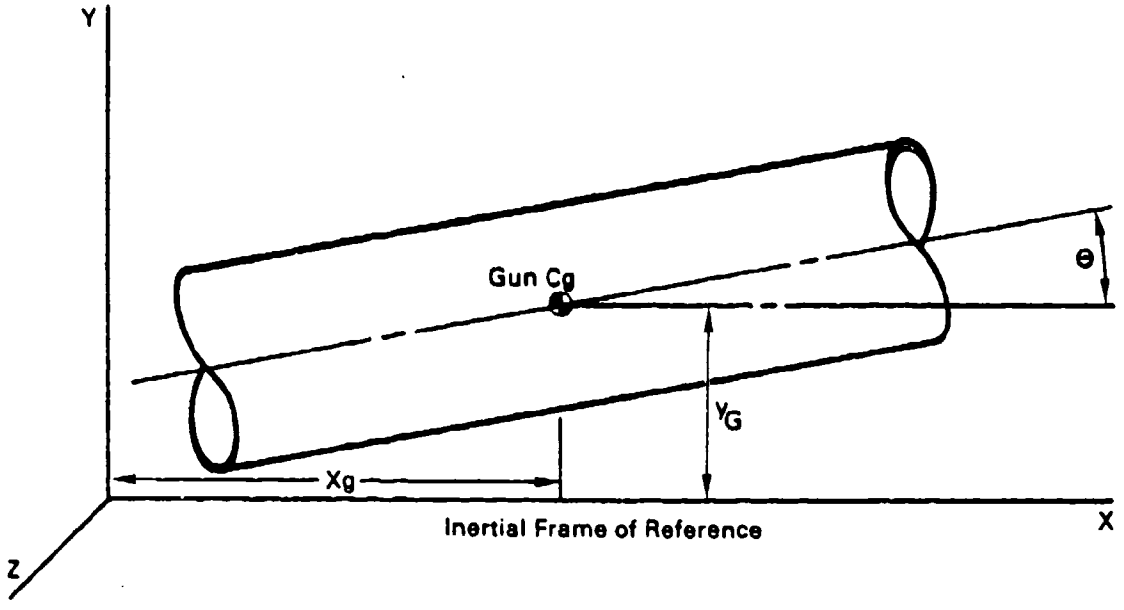

Figure 1. Displaced Position of Gun Bore

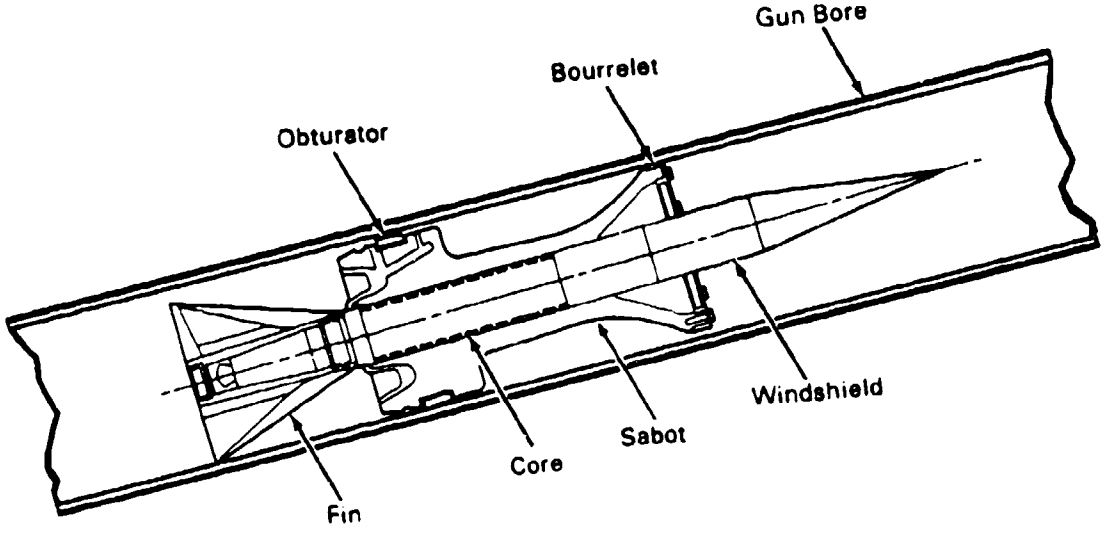

Figure 2. Projectile with Bourrelet and Obturator within Gun Bore

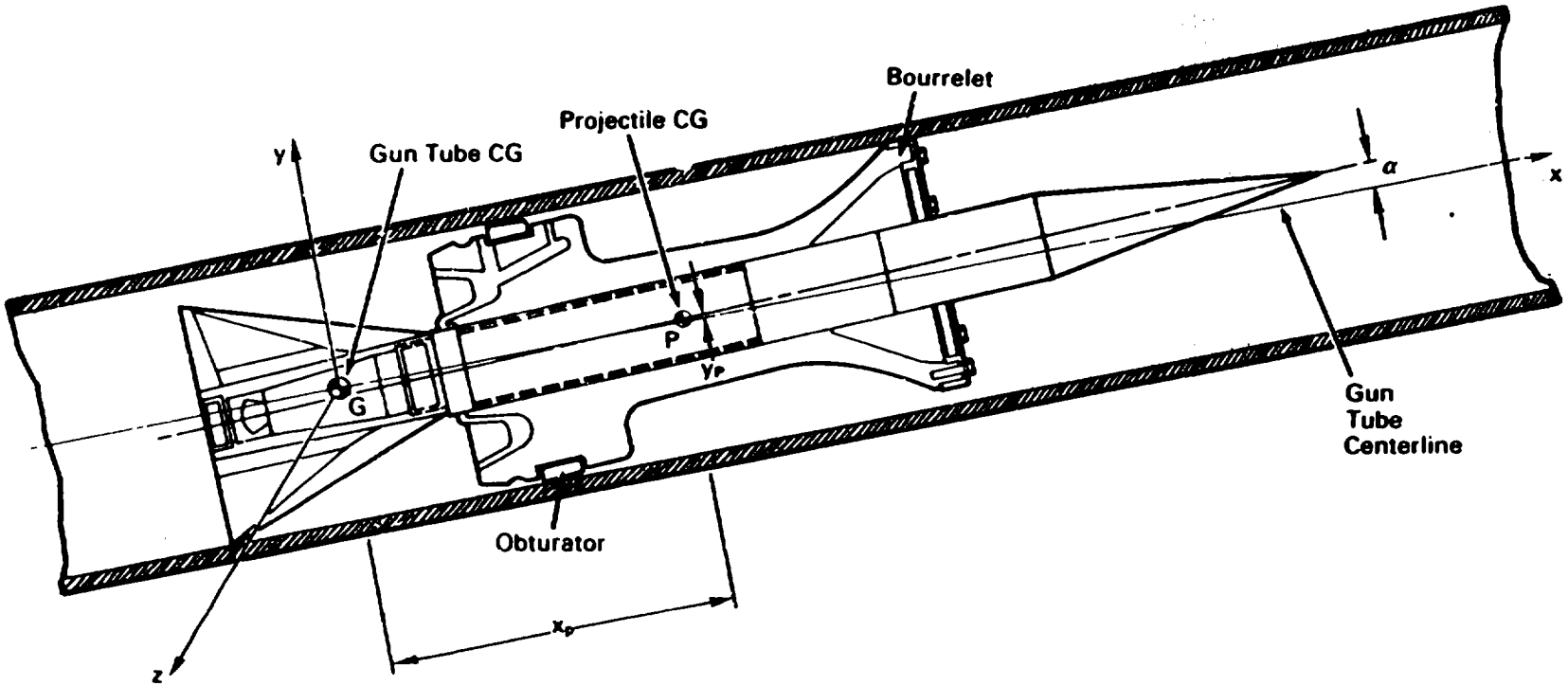

Figure 3. Body-Fixed Axes and Projectile Displacements

V. BOURRELET/GUN-BORE CONTACT MODEL

When the projectile contacts the gun bore, the rebound force at the bourrelet can be included with the help of a simple spring deflection model, as shown in Figure 4, representing the stiffness characteristics of the metallic part of the bourrelet which comes into contact with the gun tube upon impact. In this situation, the contribution to the potential energy of the system would be

$$V_{bc} = \tfrac{1}{2} k_{bc} \delta_{bc}^2 \tag{2}$$

where k_{bc} is the appropriate spring constant and δ_{bc} represents the displacement of the projectile into the gun bore, that is, the distance from the bore wall to the undistorted location of the bourrelet as determined by the orientation of the projectile axis. This displacement is (see Appendix B)

$$\delta_{bc} = \left| y_p + \ell_p \sin \alpha \right| - R_{cl,bc} \tag{3}$$

where ℓ_p represents the distance between the projectile CG and the plane of the bourrelet and $R_{cl,bc}$ is the radial clearance between the bourrelet and the bore. Contact between the bourrelet and the bore occurs when

$$\delta_{bc} \geq 0 \tag{4}$$

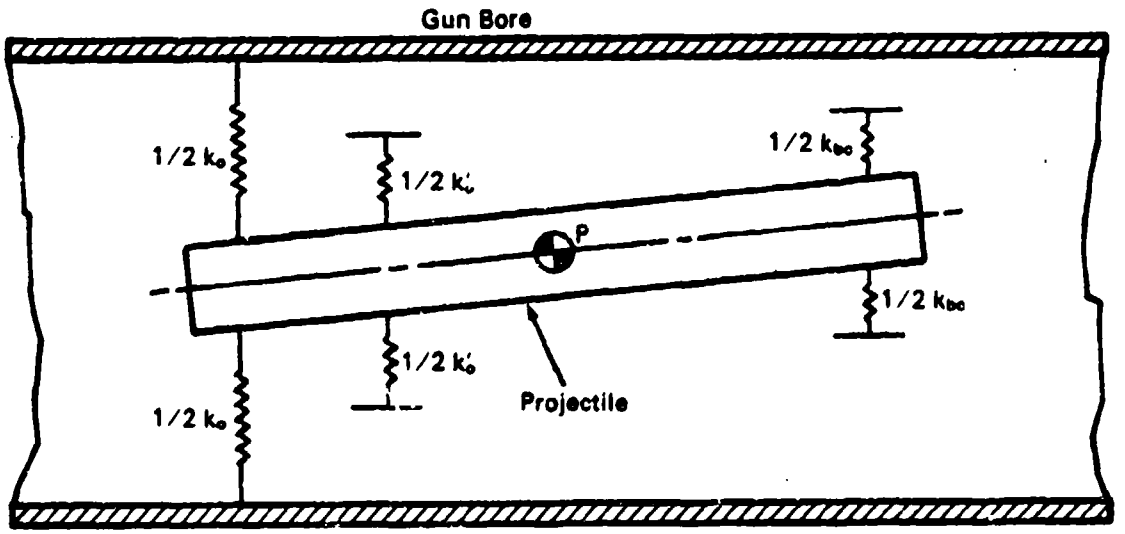

Figure 4. Obturator and Bourrelet/Gun-Bore Contact Models

This contact also contributes a friction force applied to the projectile in the negative x direction, its magnitude being

$$F_{f_{bc}} = \mu_{bc} k_{bc} \delta_{bc} \tag{5}$$

where μ_{bc} is the coefficient of friction for the bourrelet/gun-bore contact.

VI. OBTURATOR MODEL

An obturator model can be built into the balloting analysis by including transverse springs as shown in Figure 4, in the plane of the obturator, simulating the properties inherent to it. The spring k_o represents the stiffness of the plastic band while k_o' represents the stiffness of the metallic part of the obturator which should be included in the obturator stiffness when the metal impacts the gun bore. The restoring forces in these springs will tend to bring back the projectile CG towards the centerline of the gun bore in its motion down the barrel. The spring stiffness values are to be obtained either experimentally or analytically. The potential energy contribution due to these transverse springs will be

$$V_{obt} = \tfrac{1}{2} k_o \delta_o^2 \text{ (no impact)}$$
$$V_{obt} = \tfrac{1}{2} k_o \delta_o^2 + \tfrac{1}{2} k_o' (\delta_o - R_{cl,o})^2 \text{ (upon impact)} \tag{6}$$

where δ_o is the projectile displacement at the obturator, k_o and k_o' represent the stiffness of the plastic band and metallic part respectively and $R_{cl,o}$ is the radial clearance between the obturator and the bore. The displacement δ_o is given by

$$\delta_o = \left| y_p - \ell_o \sin \alpha \right| \tag{7}$$

where ℓ_o is the distance between the projectile CG and the plane of the obturator. Contact between the metallic part of the obturator and the gun bore occurs when

$$\delta_o - R_{cl,o} \geq 0 \tag{8}$$

VII. THE SYSTEM POTENTIAL ENERGY

The total potential energy of the system is given by

$$V_{System} = V_{Gun} + V_{Proj} + V_{Bourrelet\ Contact} + V_{Obt} + V_{Found\ Moment}$$

$$= m_g g Y_g + m_p g Y_p + \tfrac{1}{2} k_{bc} \delta_{bc}^2 + \tfrac{1}{2} k_o \delta_o^2 + \tfrac{1}{2} k_o' (\delta_o - R_{cl,o})^2$$

$$+ \sum_n \frac{1}{n} a_n \alpha^n \qquad (9)$$

where the third term is to be included only when there is contact between the bourrelet and the bore, the fifth term is to be included only when the metallic part of the obturator impacts the gun bore, and the last term is a power series due to the foundation moment which is a large nonlinear effect generated by the projectile as it wobbles down the gun bore and which can be measured experimentally. The quantities m_g and m_p represent the gun mass and the projectile mass respectively and n denotes the number of terms in the power series representation of the foundation moment, which is the resistance to wobbling provided by the stiffness of the obturator drive band.

VIII. EQUATIONS OF MOTION

Lagrange's equations of motion are[9]

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_r}\right) - \frac{\partial L}{\partial q_r} = Q_{r_{nc}} \quad , \quad (r = 1, 2, \ldots 6) \qquad (10)$$

where L is the Lagrangian function, $Q_{r_{nc}}$ is the non-conservative generalized force acting in each coordinate direction q_r, and $\dot{q}_r$ denotes the generalized velocity associated with the generalized coordinate q_r. The Lagrangian L is the difference between the system kinetic and potential energies, and is, in this case

---

[9] *L. Meirovitch, "Analytical Methods in Vibrations," Macmillan, New York, 1969, pp. 30-50.*

$$L = T-V = \{\tfrac{1}{2} m_g (\dot{X}_g^2 + \dot{Y}_g^2) + \tfrac{1}{2} I_g \dot{\theta}^2$$
$$+ \tfrac{1}{2} m_p(\dot{X}_p^2 + \dot{Y}_p^2) + \tfrac{1}{2} I_p \dot{\alpha}^2\}$$
$$- \{m_g g Y_g + m_p g Y_p + \tfrac{1}{2} k_{bc} \delta_{bc}^2 + \tfrac{1}{2} k_o \delta_o^2$$
$$+ \tfrac{1}{2} k_o' (\delta_o - R_{cl,o})^2 + \sum_n \frac{1}{n} a_n \alpha^n\} \qquad (11)$$

where the quantities $\dot{X}_p$ and $\dot{Y}_p$ are derived in Appendix A, δ_{bc} is given in Appendix B and δ_o and $R_{cl,o}$ are as defined in the section on Obturator Model.

Application of Lagrange's equations yields the following nonlinear, coupled equations of motion for the projectile/gun-bore system

$$q_1 = X_g: \quad (m_g + m_p) \ddot{X}_g - m_p (x_p \sin\theta + y_p \cos\theta) \ddot{\theta} + m_p \ddot{x}_p \cos\theta$$
$$- m_p \ddot{y}_p \sin\theta = m_p x_p \dot{\theta}^2 \cos\theta + 2m_p \dot{x}_p \dot{\theta} \sin\theta - m_p y_p \dot{\theta}^2 \sin\theta$$
$$+ 2 m_p \dot{y}_p \dot{\theta} \cos\theta \qquad (12)$$

$$q_2 = Y_g: \quad (m_g + m_p) \ddot{Y}_g + m_p (x_p \cos\theta - y_p \sin\theta) \ddot{\theta} + m_p \ddot{x}_p \sin\theta$$
$$+ m_p \ddot{y}_p \cos\theta = m_p x_p \dot{\theta}^2 \sin\theta - 2m_p \dot{x}_p \dot{\theta} \cos\theta + m_p y_p \dot{\theta}^2 \cos\theta$$
$$+ 2 m_p \dot{y}_p \dot{\theta} \sin\theta - (m_p + m_g) g \qquad (13)$$

$$q_3 = \theta: \quad m_p(x_p \sin\theta + y_p \cos\theta) \ddot{X}_g + m_p (x_p \cos\theta - y_p \sin\theta) \ddot{Y}_g$$
$$+ (I_g + m_p x_p^2 + m_p y_p^2) \ddot{\theta} - m_p y_p \ddot{x}_p + m_p x_p \ddot{y}_p$$
$$+ m_p \{2 \dot{\theta} (x_p \dot{x}_p + y_p \dot{y}_p) - \dot{X}_g [(\dot{x}_p \sin\theta + \dot{y}_p \cos\theta)$$
$$+ \dot{\theta} (x_p \cos\theta - y_p \sin\theta)] + \dot{Y}_g [(\dot{x}_p \cos\theta - \dot{y}_p \sin\theta)$$
$$- \dot{\theta} (x_p \sin\theta + y_p \cos\theta)]\} - m_p \{[\dot{X}_g y_p - \dot{Y}_g x_p) \dot{\theta}$$
$$- (\dot{X}_g \dot{x}_p + \dot{Y}_g \dot{y}_p)] \sin\theta - [(\dot{X}_g x_p + \dot{Y}_g y_p) \dot{\theta}$$
$$+ (\dot{Y}_g \dot{x}_p - \dot{X}_g \dot{y}_p)] \cos\theta\} = 0 \qquad (14)$$

$$q_4 = x_p: \; m_p \{\ddot{X}_g + \ddot{Y}_g + \ddot{\theta}\,[x_p(\cos\theta - \sin\theta) - y_p(\sin\theta + \cos\theta)]$$

$$+ (\sin\theta + \cos\theta)\,\ddot{x}_p + (\cos\theta - \sin\theta)\,\ddot{y}_p - 2\,\dot{y}_p\dot{\theta}\,(\sin\theta + \cos\theta)$$

$$+ 2\,\dot{x}_p\dot{\theta}\,(\cos\theta - \sin\theta) - (x_p + y_p)\,\dot{\theta}^2\cos\theta + (y_p - x_p)\,\dot{\theta}^2\sin\theta$$

$$- x_p\,\dot{\theta}^2 - \dot{y}_p\,\dot{\theta} - (\dot{Y}_g\cos\theta - \dot{X}_g\sin\theta)\,\dot{\theta} + g\sin\theta\}$$

$$= f_x(t) - \mu_{bc}\,k_{bc}(\left|y_p + \ell_p\,\alpha\right| - R_{c1,bc})\,\mathrm{sgn}\,(\dot{x}_p) \qquad (15)$$

$$q_5 = y_p: \; m_p \{-\ddot{X}_g\sin\theta + \ddot{Y}_g\cos\theta + x_p\,\ddot{\theta} + \ddot{y}_p + 2\,\dot{x}_p\,\dot{\theta} - \dot{Y}_g\,\dot{\theta}\sin\theta$$

$$- \dot{X}_g\,\dot{\theta}\cos\theta - y_p\dot{\theta}^2 + \dot{\theta}\,(\dot{X}_g\cos\theta + \dot{Y}_g\sin\theta)$$

$$+ g\cos\theta\} + k_{bc}\,(y_p + \ell_p\,\alpha - R_{c1,bc}\,\mathrm{sgn}\,(y_p + \ell_p\,\alpha))$$

$$+ (k_o + k_o')\,(y_p - \ell_o\alpha) - R_{c1,o}\;k_o'\,\mathrm{sgn}\,(y_p - \ell_o\,\alpha) = f_y(t) \qquad (16)$$

$$q_6 = \alpha: \; I_p\,\ddot{\alpha} + k_{bc}\,(y_p + \ell_p\alpha - R_{c1,bc}\,\mathrm{sgn}\,(y_p + \ell_p\,\alpha))\,\ell_p$$

$$- (k_o + k_o')\,\ell_o\,(y_p - \ell_o\,\alpha)$$

$$+ R_{c1,o}\,k_o'\,\ell_o\,\mathrm{sgn}\,(y_p - \ell_o\alpha) + \sum_n a_n\,\alpha^{n-1} + c_1\dot{\alpha} + c_2\,\dot{\alpha}\left|\dot{\alpha}\right| = m\,(t) \qquad (17)$$

Equations (12-17) can be written down in the form

$$[M]\{\ddot{q}\} + [C]\,\{\dot{q}\} + [K]\,\{q\} = \{F\} \qquad (18)$$

where [M], [C], and [K] are 6 x 6 matrices, {F} is a force column matrix and {q} is a vector of generalized coordinates. The elements of these matrices are as follows:

$M_{11} = (m_g + m_p)$, $M_{13} = -\,m_p\,(x_p\sin\theta + y_p\cos\theta)$,

$M_{14} = m_p\cos\theta$, $M_{15} = -\,m_p\sin\theta$,

$M_{22} = (m_g + m_p)$, $M_{23} = m_p\,(x_p\cos\theta - y_p\sin\theta)$,

$M_{24} = m_p\sin\theta$, $M_{25} = m_p\cos\theta$,

$M_{31} = -\,m_p\,(x_p\sin\theta + y_p\cos\theta)$,

$M_{32} = m_p\,(x_p\cos\theta - y_p\sin\theta)$,

$$M_{33} = I_g + m_p(x_p^2 + y_p^2) \ , \ M_{34} = - m_p y_p \qquad (19)$$

$$M_{35} = m_p x_p \ , \ M_{41} = m_p \ , \ M_{42} = m_p \ ,$$

$$M_{43} = m_p \left[x_p (\cos\theta - \sin\theta) - y_p (\sin\theta + \cos\theta)\right] \ ,$$

$$M_{44} = m_p (\sin\theta + \cos\theta) \ , \ M_{45} = m_p (\cos\theta - \sin\theta) \ ,$$

$$M_{51} = - m_p \sin\theta \ , \ M_{52} = m_p \cos\theta \ ,$$

$$M_{53} = m_p x_p \ , \ M_{55} = m_p \ , \ M_{66} = I_p \ .$$

$$C_{14} = -2 m_p \dot{\theta} \sin\theta \ , \ C_{15} = -2 m_p \dot{\theta} \cos\theta \ ,$$

$$C_{23} = 2 m_p (\dot{x}_p \cos\theta - \dot{y}_p \sin\theta) \ ,$$

$$C_{31} = - \{(\dot{x}_p \sin\theta + \dot{y}_p \cos\theta) + \dot{\theta} (x_p \cos\theta - y_p \sin\theta)\} m_p \ ,$$

$$C_{32} = \{(\dot{x}_p \cos\theta - \dot{y}_p \sin\theta) - \dot{\theta} (x_p \sin\theta + y_p \cos\theta)\} m_p \ ,$$

$$C_{33} = - m_p \{(\dot{X}_g y_p - \dot{Y}_g x_p) \sin\theta - (\dot{X}_g x_p + \dot{Y}_g y_p) \cos\theta\} \ ,$$

$$C_{34} = m_p (\dot{X}_g \sin\theta - \dot{Y}_g \cos\theta) \ ,$$

$$C_{35} = m_p (\dot{Y}_g \sin\theta + \dot{X}_g \sin\theta) \ ,$$

$$C_{41} = m_p \dot{\theta} \sin\theta \ , \ C_{42} = - m_p \dot{\theta} \cos\theta \ , \qquad (20)$$

$$C_{43} = - m_p (x_p \dot{\theta} + \dot{y}_p) \ , \ C_{44} = 2 m_p \dot{\theta} (\cos\theta - \sin\theta) \ ,$$

$$C_{45} = -2\dot{\theta} m_p (\sin\theta + \cos\theta) \ ,$$

$$C_{51} = -m_p \dot{\theta} \cos\theta \ , \ C_{52} = -m_p \dot{\theta} \sin\theta \ ,$$

$$C_{53} = m_p (\dot{X}_g \cos\theta + \dot{Y}_g \sin\theta) \ , \ C_{54} = 2 m_p \dot{\theta} \ ,$$

$$C_{66} = c_1 + c_2 \ |\dot{\alpha}| \ .$$

$$K_{14} = -m_p \dot{\theta}^2 \cos\theta \ , \ K_{15} = m_p \dot{\theta}^2 \sin\theta \ ,$$

$$K_{24} = -m_p \dot{\theta}^2 \sin\theta \ , \ K_{25} = -m_p \dot{\theta}^2 \cos\theta \ ,$$

$$K_{34} = 2m_p \dot{\theta} \dot{x}_p \ , \ K_{35} = 2m_p \dot{\theta} \dot{y}_p \ ,$$

$$K_{44} = -m_p \dot{\theta}^2 (\cos\theta + \sin\theta) \ , \ K_{45} = m_p \dot{\theta}^2 (\sin\theta - \cos\theta) \ ,$$

$$K_{55} = -m_p \dot{\theta}^2 + k_{bc} + k_o + k_o' \ , \tag{21}$$

$$K_{56} = k_{bc} \ell_p - (k_o + k_o') \ell_o \ ,$$

$$K_{65} = k_{bc} \ell_p - (k_o + k_o') \ell_o \ ,$$

$$K_{66} = k_{bc} \ell_p^2 + (k_o + k_o') \ell_o^2 + \sum_n a_n \alpha^{n-2} \ .$$

$$F_2 = -(m_p + m_g) g \ ,$$

$$F_4 = -m_p g \sin\theta + f_x(t) - \mu_{bc} k_{bc} \left(\left| y_p + \ell_p \alpha \right| - R_{cl,bc} \right) \operatorname{sgn}(\dot{x}_p) \ , \tag{22}$$

$$F_5 = -m_p g \cos\theta + k_{bc} R_{cl,bc} \operatorname{sgn}(y_p + \ell_p \alpha) + f_y(t) + R_{cl,o} k_o' \operatorname{sgn}(y_p - \ell_o \alpha) \ ,$$

$$F_6 = m(t) + k_{bc} R_{cl,bc} \ell_p \operatorname{sgn}(y_p + \ell_p \alpha) - R_{cl,o} k_o' \ell_o \operatorname{sgn}(y_p - \ell_o \alpha) \ .$$

All other elements are zero.

IX. SOLUTION TECHNIQUE

The equations of motion (Equations 18) for the projectile/gun tube system are coupled, nonlinear differential equations which can be solved only numerically. If the values of q's and $\dot{q}$'s are known at any instant, their substitution in [M], [C], [K] and {F} will lead to a set of 6 simultaneous algebraic equations with the q's as the unknowns. Elements of these matrices are first computed from given initial values of q's and $\dot{q}$'s at the first time step and then the simultaneous equations generated are solved by Gaussian elimination

to find the $\ddot{q}$'s which lead to the solution for the subsequent time step. Essentially, we have a set of six coupled nonlinear differential equations to be solved at each time step. The solution to these equations may be obtained by Newmark's constant average-acceleration method of integration[10] which is a self-starting technique that yields a step-by-step solution up to a specified time limit, once the initial positions and velocities are supplied. The time step can be monitored during the process of computation if needed. At impact and rebound between the projectile and the gun bore, either at the bourrelet or at the obturator, the solution of the equations of motion need not be interrupted although additional forces will come into play from the activation of springs k_{bc} and k_0' and from the frictional effect between the bourrelet and the bore when the bourrelet slides on the walls of the gun tube. The projectile/gun tube dynamic system response can then be studied at any desired time and time histories of all coordinates describing the system behavior can easily be obtained.

The first consideration in selecting a numerical integration scheme for multi degree of freedom systems, as Craig[11] points out, should be its stability. In most cases, it is desirable to use a method that is unconditionally stable such as the constant-average-acceleration method. This method also produces no amplitude error, that is, there is no numerical dissipation, regardless of the time step size. The period error, too, is small. However, for some multi degree of freedom systems, it is desirable to have numerical dissipation to filter out the response of less accurate higher modes, which is similar to truncating higher modes in a mode-superposition solution. According to Craig, Newmark's linear acceleration method has an advantage over his constant-average-acceleration method in that it provides this damping for higher modes. One difficulty, however, is that it is not unconditionally stable. In order to ensure an accurate solution to the balloting projectile problem, both of these methods were resorted to and, for small time steps, no

---

[10] *K.J. Bathe and E.L. Wilson, "Numerical Methods in Finite Element Analysis," Prentice Hall, Englewood Cliffs, N.J., 1976, pp. 322-325.*

[11] *R.R. Craig, Jr., "Structural Dynamics - An Introduction to Computer Methods," John Wiley, New York, 1981, pp. 461-463.*

difference in the results was noted. It was therefore decided to use the constant-average-acceleration method for this work because of its unconditional stability.

X. COMPUTER CODE DESCRIPTION

All computer code used in this study was written in Pascal on the Hewlett-Packard 9816 Series 200 Desk Top Computer. The main program, NEWMARK, performs numerical integration based on either the linear acceleration or constant-average-acceleration method. MOPBOX (Matrix OPerations tool BOX) provides various low-level routines to manipulate matrix and vector entities, including addition, subtraction, multiplication, etc. There is also a routine that solves linear algebraic equations by Gauss-Jordan Elimination. Graphics support is provided by the PLOTPAC Graphic Plotting Package. For a complete listing of the code, see Appendix D - Computer Program Listing.

NEWMARK, as well as its supporting modules, MOPBOX and PLOTPAC, were tested using the example of a linear, two degree of freedom system given in Bathe & Wilson[10] on pp. 324-325. The input used is shown on page 76, Appendix D. Results from NEWMARK were in exact agreement with those given in this reference.

XI. NUMERICAL RESULTS AND DISCUSSION

The balloting projectile model and analysis developed have been utilized to simulate the dynamical behavior of an APFSDS-T (Armor Piercing Fin Stabilized Discarding Sabot with a Tracer) projectile in a 120 mm M1A1 Tank Main Gun Tube in the following situations:

Case 1 — Projectile CG at the Obturator ($\ell_0 = 0$)

(a) Initial condition: $y_{obt} = 0$, $\alpha = 0.054$ deg , corresponding to a maximum tilt of the projectile with no displacement at the obturator.

(b) Initial condition: $y_{obt} = -.0026$ in , $\alpha = 0.054$ deg , corresponding to same projectile tilt as in (a) in conjunction with a displacement at the obturator equal to one-half of the radial clearance between the projectile and the gun bore at the obturator.

Case 2 — Projectile CG Between the Obturator and the Bourrelet, 2 inches from Obturator ($\ell_o = 2$ inches)

(a) Same as in 1(a).

(b) Same as in 1(b).

Other data used is as follows:

Gun weight	= 4000 lbs
Projectile weight	= 16 lbs
Projectile cross-sectional area	= 17.53 in^2
I_p	= 0.583 lb-in-sec^2
I_g	= 150.0 lb-in-sec^2
c_1	= 71 lb-in-sec, approximately 5% of critical
c_2	= 0
k_{bc}	= 5.0×10^7 lbs/in
k_o	= 5.0×10^6 lbs/in
k_o'	= 1.0×10^8 lbs/in
ℓ_p	= 5.5 inches
$R_{cl, bc}$	= 0.0052 inches
$R_{cl, o}$	= 0.0052 inches
μ_{bc}	= 0

Time history of gas pressure p(t) on the projectile as given in Figure 5. The curve for p(t) shown is a modification that results from inclusion of the effect due to friction at the obturator/bore interface.

Projectile foundation moment as given in Figure 6.

The time step used for all solutions is 0.005 msec.

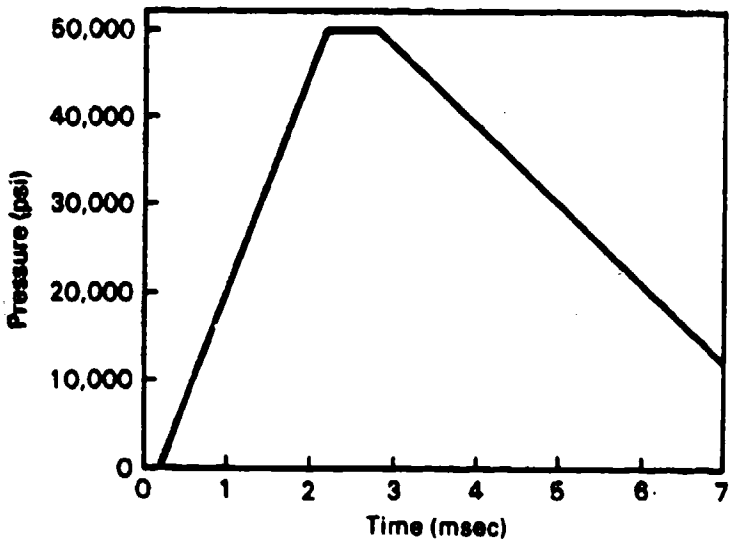

Figure 5. Time History of Gas Pressure on Projectile

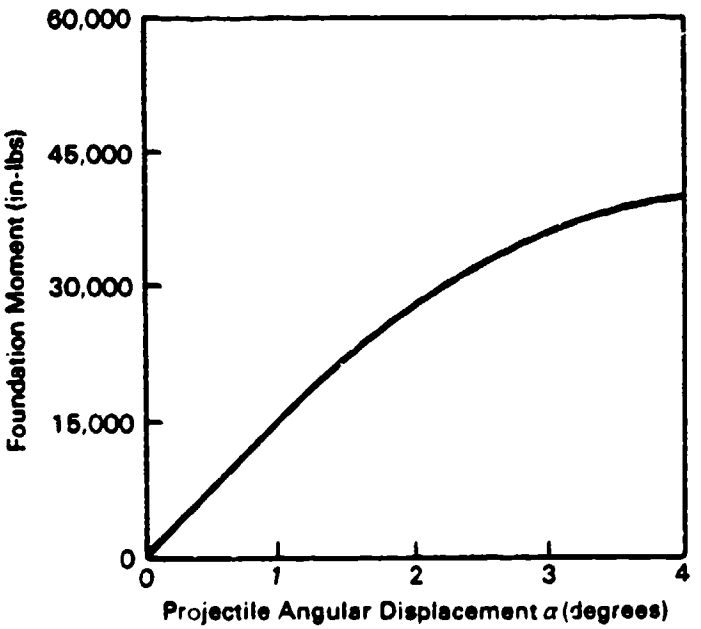

Figure 6. Projectile Foundation Moment

Gun Horizontal and Vertical Displacements and Projectile x-Displacement Response

The time histories of the horizontal gun displacement $X_g(t)$, the vertical gun displacement $Y_g(t)$ and the x-displacement $x_p(t)$ of the projectile generated by the analysis are shown in Figures 7-9. These are similar for all the cases analyzed and no significant changes are noted from one case to the next.

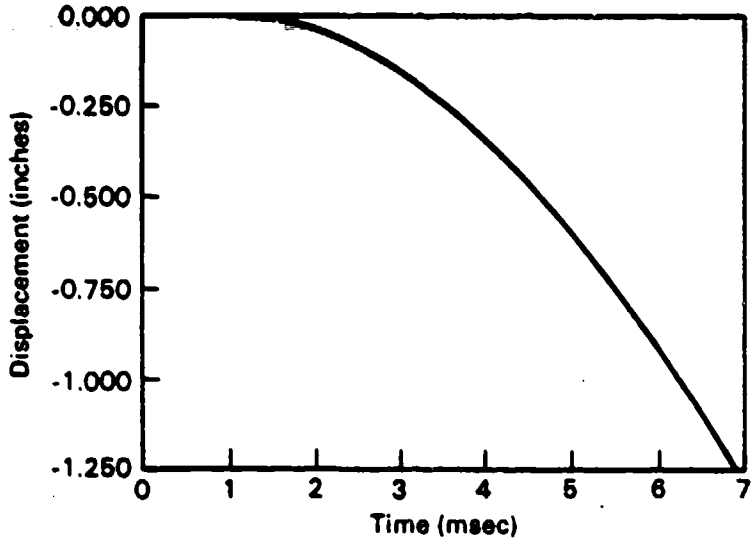

Figure 7. Time History of Horizontal Gun Displacement (X_g)

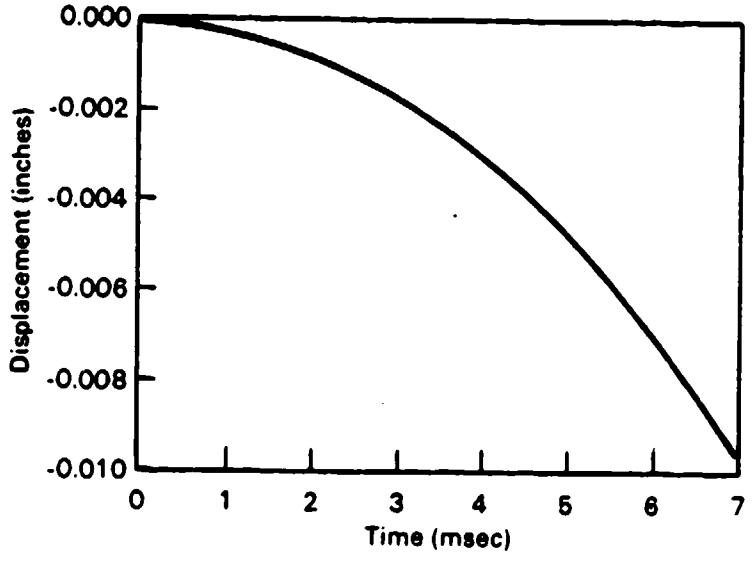

Figure 8. Time History of Vertical Gun Displacement (Y_g)

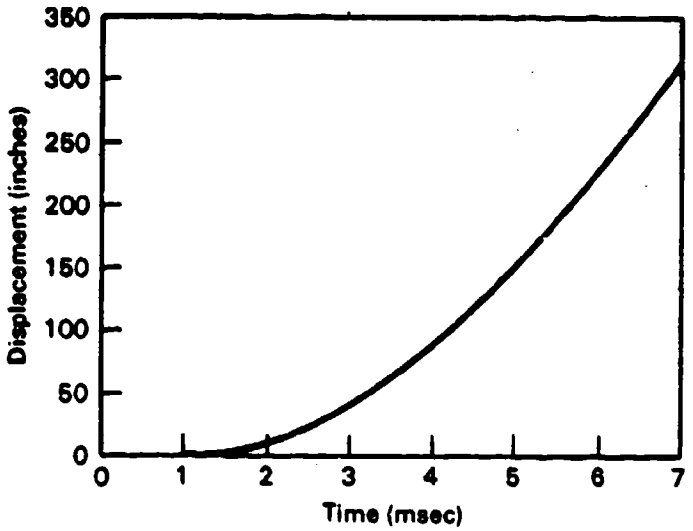

Figure 9. Time History of X-Displacement of Projectile (x_p)

Gun Angular Displacement Response

The time histories of the angular displacement $\theta(t)$ of the gun for the various cases are represented in Figures 10-13. For a projectile with its CG located at the obturator, and initially cocked in the bore with no displacement at the obturator, the angular gun displacement $\theta(t)$ gradually increases with time as shown in Figure 10 until the projectile exits the bore with no impact occurring between the bourrelet and the bore at any time at all. When an initial vertical displacement is imposed at the obturator, oscillations of the tube result from continual impact between the bourrelet and the bore as shown in Figure 11. For the case in which the projectile CG is located away from the obturator and the projectile is initially cocked in the bore with no displacement at the obturator, the gun angle increases until the first impact between the bourrelet and the bore and then begins to oscillate as seen in Figure 12. With an additional initial vertical displacement imposed at the obturator, the oscillations are seen to be small until the first moment of impact between the bourrelet and the bore, following which oscillation amplitudes increase with further impacts.

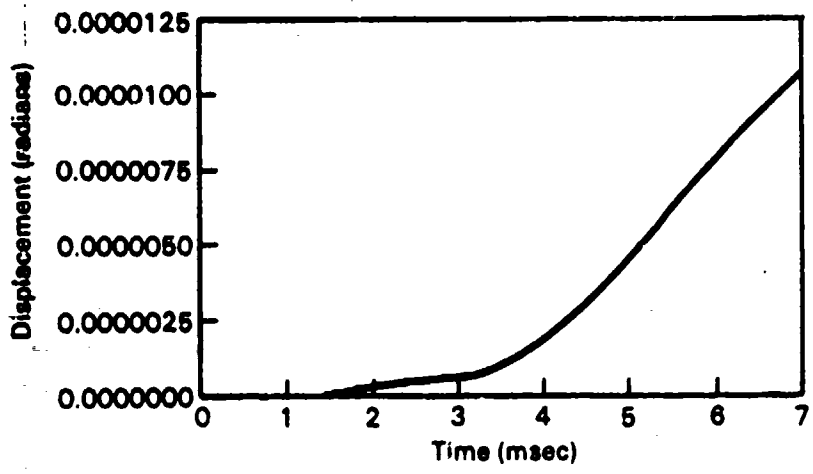

Figure 10. Time History of Gun Angular Displacement (θ) for Case 1a

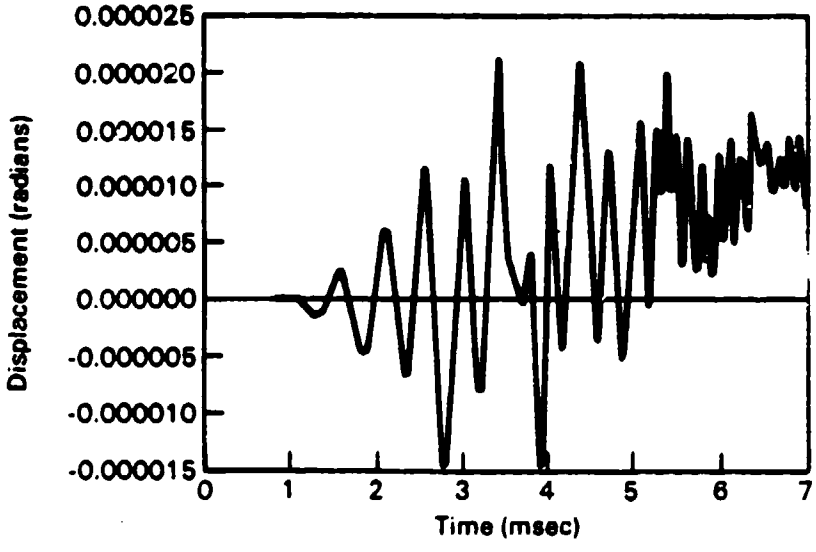

Figure 11. Time History of Gun Angular Displacement (θ) for Case 1b

Projectile y-Displacement Response

For case 1(a), Figure 14 essentially reveals a low-frequency type oscillatory behavior that results from the projectile not impacting the gun bore at all. For case 1(b), owing to continual impact between the bourrelet and the bore, the oscillations are seen to be of higher frequency as shown in Figure 15. For case 2(a), Figure 16 shows a gradual fall of displacement until the occurrence of the first impact between the bourrelet and the bore leading to

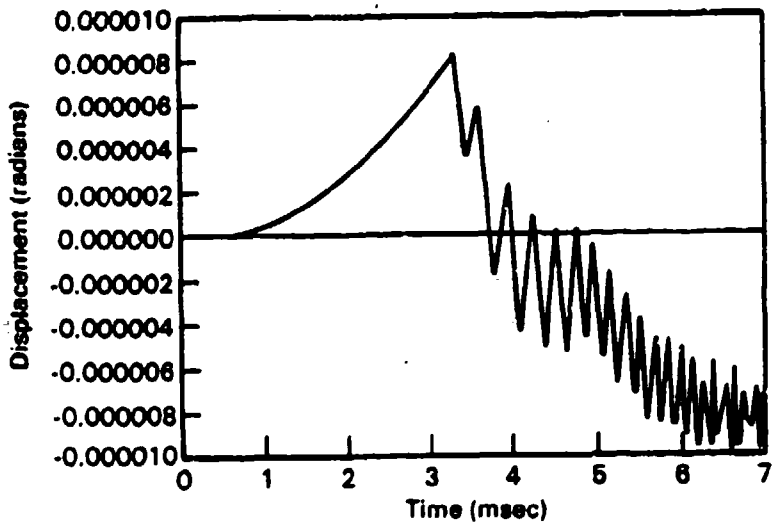

Figure 12. Time History of Gun Angular Displacement (θ) for Case 2a

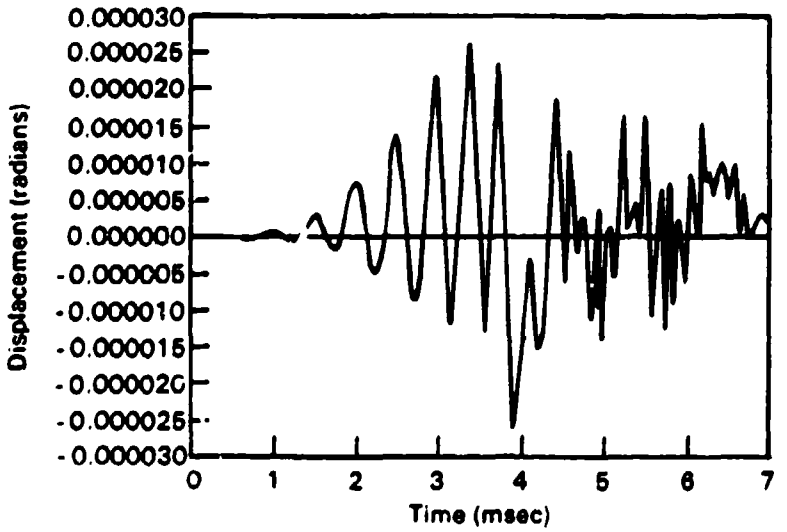

Figure 13. Time History of Gun Angular Displacement (θ) for Case 2b

the subsequent oscillatory behavior pattern depicted. For case 2(b) (see Figure 17) the behavior is seen to be a sinusoidal motion superimposed upon an almost linear decay of amplitude until the first impact, after which time, an increase in frequency occurs.

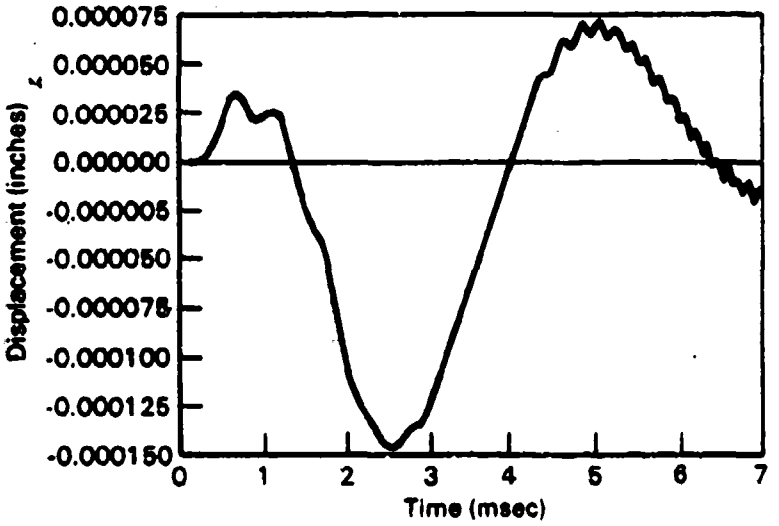

Figure 14. Time History of Y-Displacement of Projectile (y_p) for Case 1a

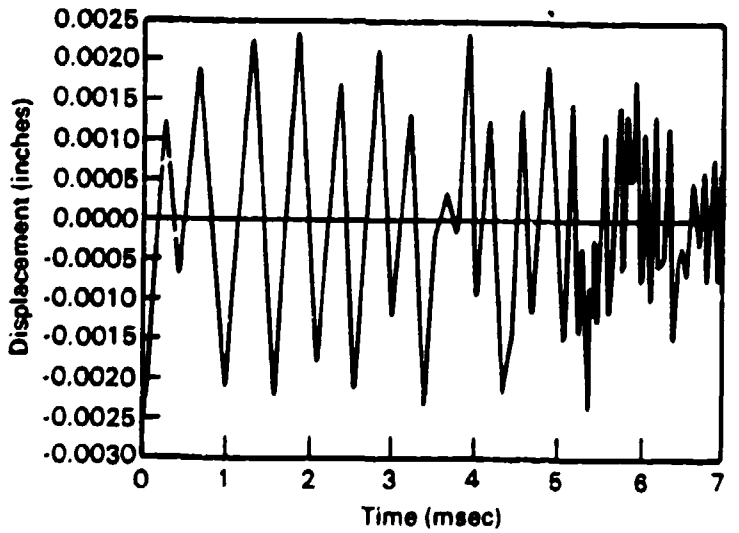

Figure 15. Time History of Y-Displacement of Projectile (y_p) for Case 1b

Projectile Yaw Response

The time histories of the yawing motion of the projectile $\alpha(t)$ generated for the various cases are represented in Figures 18-21. For case 1(a), the response is a low frequency, sinusoidal type of motion as shown in Figure 18, which results from the projectile not impacting the gun tube at all. For case 1(b) (see Figure 19), because of continual impact between the bourrelet and the bore, oscillations of larger frequency occur. For case 2(a) (see Figure

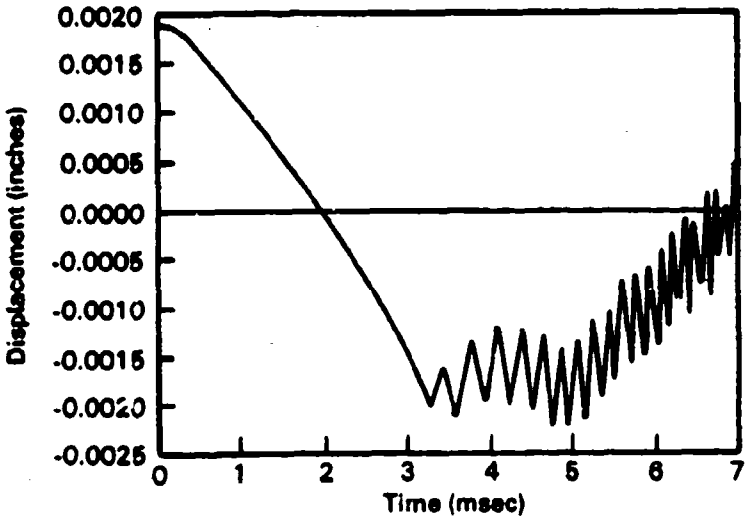

Figure 16. Time History of Y-Displacement of Projectile (y_p) for Case 2a

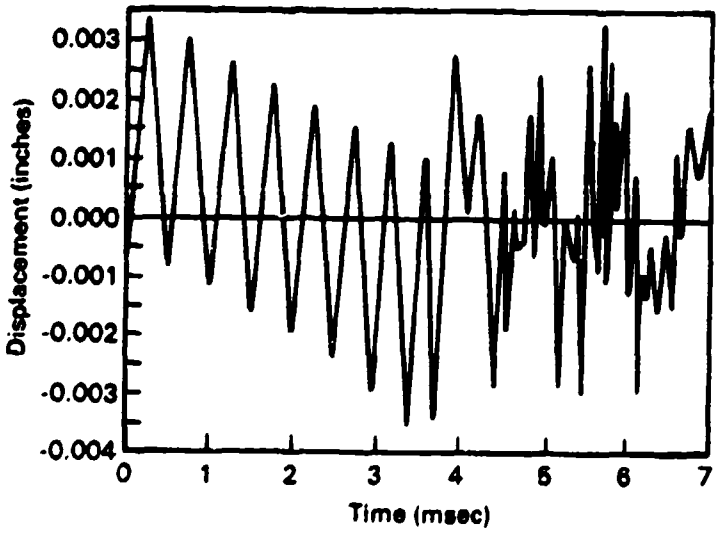

Figure 17. Time History of Y-Displacement of Projectile (y_p) for Case 2b

20), the bourrelet is initially in contact at the top of the bore. As the projectile travels down the gun barrel, the bourrelet drops making impact twice before exiting. For case 2(b) (see Figure 21), pre-impact oscillations are relatively small, but become large following bourrelet contact.

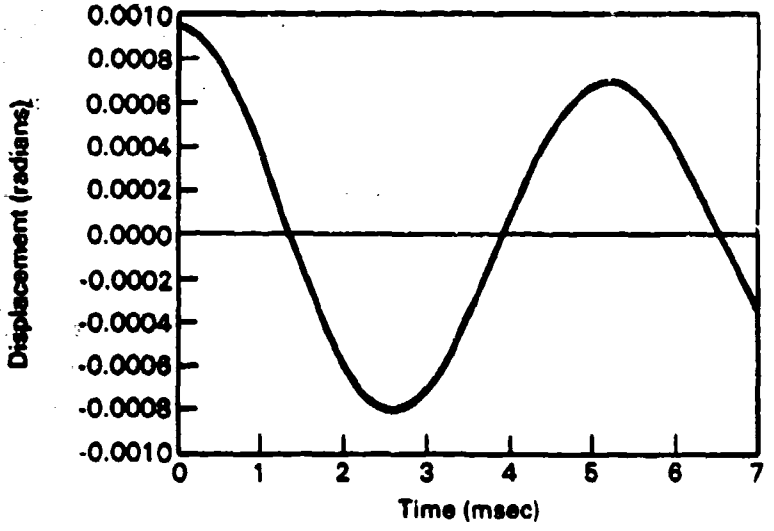

Figure 18. Time History of Yawing Motion of Projectile (α) for Case 1a

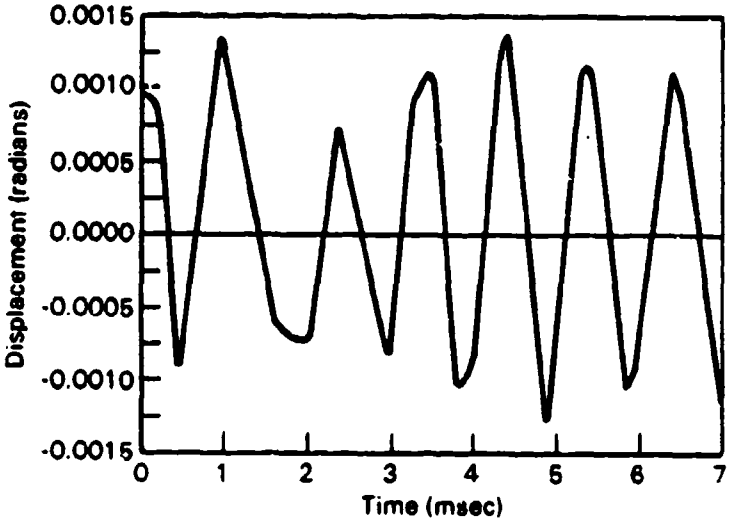

Figure 19. Time History of Yawing Motion of Projectile (α) for Case 1b

Effect of Using a Smaller Time Step for Integration

In order to determine the effect of a smaller time step on the dynamic behavior of the system, case 2a is also analyzed using Δt = .0005 msec. There is no appreciable change in results except that the oscillation period becomes slightly smaller, which is in agreement with Craig's observation[11].

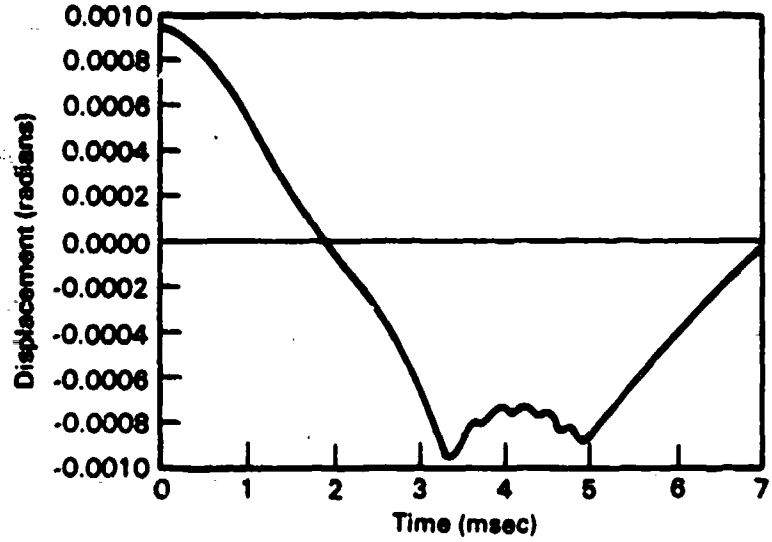

Figure 20. Time History of Yawing Motion of Projectile (α) for Case 2a

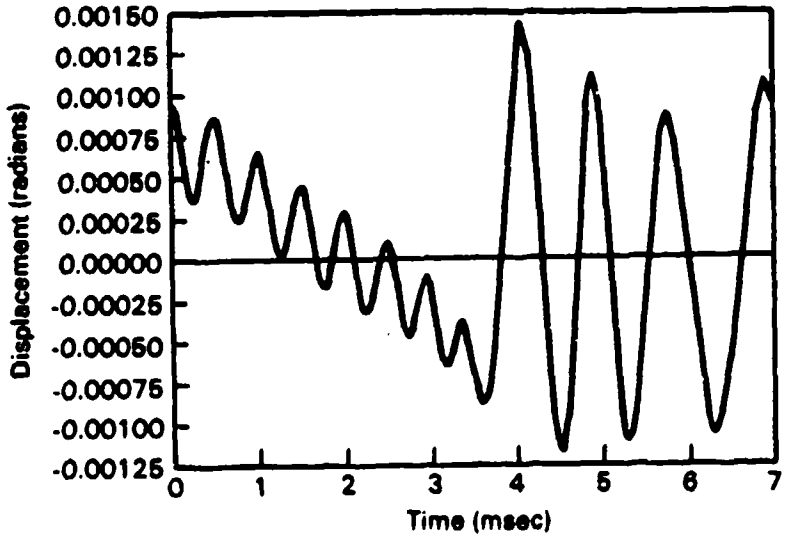

Figure 21. Time History of Yawing Motion of Projectile (α) for Case 2b

Other Observations

The results shown indicate that the projectile motion oscillates about the limits set by the clearance between the projectile bourrelet and the gun bore. The maximum longitudinal deceleration of the gun is 220 g's and occurs at about 2 msec for all the cases analyzed. The maximum lateral acceleration or deceleration of the gun is lowest (1 g) for Case 1a, in which the projectile CG is at the obturator and the projectile is initially cocked in the bore with

no vertical displacement at the obturator. It is largest (14 g) for Case 2b, in which the projectile CG is 2 inches away from the obturator and the projectile is initially cocked in the bore with a vertical displacement at the obturator. The maximum projectile longitudinal acceleration is seen to be 57,000 g, the same for all cases. The maximum projectile lateral acceleration or deceleration is the lowest (45 g) for Case 1a and highest (48,000 g) for Case 2b. The maximum projectile yaw rate is the lowest (0.8 rads/sec) for Case 2a and highest (9.5 rads/sec) for Case 2b, with the maximum yaw being of the order of .001 radians for all cases. Details are shown in Table 1. In general, the projectile whose CG is situated away from the obturator and which is initially cocked in the bore with a prescribed vertical displacement at the obturator (Case 2b) seems to exhibit the most vigorous dynamic behavior, as expected.

For the projectile whose CG is at the obturator and which is initially cocked in the bore but has no prescribed initial vertical displacement at the obturator, the results indicate that there is no impact at all between the bourrelet and the bore throughout the in-bore period. Consequently, Equation (17) which is the equation of motion corresponding to the sixth generalized coordinate, namely, the projectile yawing motion, reduces to an uncoupled second order differential equation in the coordinate α. If the foundation moment curve is linear over the range of the in-bore period, as is the case here, this equation of motion further reduces to a linear uncoupled second order differential equation in α. Thus, so long as the projectile CG is situated at the obturator and the obturator has no initial vertical displacement, the projectile could be assumed to be a single degree of freedom system for obtaining a good initial estimate of yawing response. However, for other situations, a more rigorous multi degree of freedom system analysis would seem necessary, as the results of this work indicate.

XII. CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER STUDY

In order to ensure accuracy in internal ballistics, a realistic prediction of the yawing motion of a projectile is necessary. A dynamic analysis incorporating a six degree of freedom mathematical model of a projectile/gun-

TABLE 1. COMPARISON OF MAXIMUM RESPONSES FOR THE DIFFERENT CASES ANALYZED

Case	Maximum Gun Longitudinal Deceleration (g's)	Maximum Gun Lateral Acceleration/ Deceleration (g's)	Maximum Projectile Longitudinal Acceleration (g's)	Maximum Projectile Lateral Acceleration/ Deceleration (g's)	Maximum Projectile Yaw (rads)	Maximum Projectile Yaw Rate (rads/sec)
1a	220	1	57,000	45	0.00095	1.0
1b	220	8	57,000	32,000	0.00125	7.0
2a	220	2.5	57,000	5,000	0.00095	0.8
2b	220	14	57,000	48,000	0.0015	9.5

tube system is presented in this report. The nonlinear differential equations of motion of the system are derived using a Lagrangian formulation with the effects of obturator flexibility and projectile impact and rebound at the bourrelet and the obturator included. Using Gaussian elimination and Newmark's constant-average-acceleration integration scheme, the nonlinear equations are solved to yield usable dynamical response information for some specific test cases.

The analysis and results indicate that as long as the CG of the projectile is located at the obturator and no initial vertical displacement is prescribed at the obturator, the projectile can be treated as a simple single degree of freedom system for estimating its yawing response. For more complex situations, however, a more rigorous analysis employing a larger number of degrees of freedom such as the one suggested in this report seems to be in order.

The following recommendations for further work are bound to constitute additional contributions to the general problem of the dynamics of a balloting projectile in a moving gun tube.

1. Account for the three-dimensional motion of the projectile and the gun employing Euler's angles.

2. Treat the projectile as a rigid body moving within an elastic gun tube.

3. Investigate the effect of considering the projectile as a flexible body.

4. Include the effect of rifling of the projectile.

ACKNOWLEDGMENT

The authors are grateful to E.M. Patton for providing insight into the physical aspects of gun dynamics.

REFERENCES

1. F.V. Reno, "The Motion of the Axis of a Spinning Shell Inside the Bore of a Gun," BRL Report No. BRL-R-320, 1943 (AD# 491839).

2. L.H. Thomas, "The Motion of the Axis of a Spinning Shell Inside the Bore of a Gun," BRL Report No. BRL-R-544, 1945 (PB# 22102).

3. J.G. Darpas, Translated by H.P. Hitchcock, "Transverse Forces on Projectiles Which Rotate in the Barrel," BRL Report No. BRL-MR-1208, 1959 (Memorial de l'artillerie francaise, 31:19, No. 1, 1957) (AD# 218873).

4. F.J. Perdreaville, "Analysis of the Lateral Motion of a Projectile in the Gun Tube," Sandia Laboratories, Albuquerque, NM, SC-RR-710071, 1971.

5. S.H. Chu and F.K. Soechting, "Transverse Motion of an Accelerating Shell," Picatinny Arsenal, Dover, NJ, PA-TR-4314, 1972 (AD# 894572).

6. F.J. Perdreaville, "Analysis of the Lateral Motion of an Unbalanced Projectile in a Rigid Gun Tube," Sandia Laboratories, Albuquerque, NM, 87115, SAND 74-0361, 1974.

7. F.J. Perdreaville, "Analysis of the Lateral Motion of an Unbalanced Projectile in an Elastic Gun Tube," Sandia Laboratories, Albuquerque, NM, 87115, SAND 74-0362, 1974.

8. H.L. Langhaar and A.P. Boresi, "Dynamics of a Projectile in a Concentric Flexible Tube," BLM Applied Mechanics Consultants, Contract Report No. ARBRL-CR-00501, February 1983.

9. L. Meirovitch, "Analytical Methods in Vibrations," Macmillan, New York, 1969, pp. 30-50.

10. K.J. Bathe and E.L. Wilson, "Numerical Methods in Finite Element Analysis," Prentice Hall, Englewood Cliffs, N.J., 1976, pp. 322-325.

11. R.R. Craig, Jr., "Structural Dynamics - An Introduction to Computer Methods", John Wiley, New York, 1981, pp. 461-463

APPENDIX A

PROJECTILE CENTER OF GRAVITY DISPLACEMENTS AND VELOCITIES

APPENDIX A

PROJECTILE CENTER OF GRAVITY DISPLACEMENTS AND VELOCITIES

The position vector $\vec{r}_p$ of the projectile center of mass in the inertial frame of reference can be written as

$$\vec{r}_p = \vec{r}_{P/G} + \vec{r}_G \tag{A-1}$$

when $\vec{r}_{P/G}$ represents the position vector of the projectile center of mass with respect to the gun tube center of mass, and $\vec{r}_G$ represents the position vector of the gun tube center of mass in the inertial frame of reference. Using $\hat{i}$ and $\hat{j}$ as unit vectors along X and Y axes and $\hat{\ell}$ and $\hat{m}$ as those along the body-fixed x and y axes, we can write

$$\vec{r}_{P/G} = x_p\hat{\ell} + y_p\hat{m}$$
$$\vec{r}_G = X_g\hat{i} + Y_g\hat{j} \tag{A-2}$$

where $\hat{\ell} = \cos\theta\,\hat{i} + \sin\theta\,\hat{j}$

$$\hat{m} = -\sin\theta\,\hat{i} + \cos\theta\,\hat{j} \tag{A-3}$$

Substituting equations (A-2) into (A-1), we get

$$\vec{r}_p = X_p\hat{i} + Y_p\hat{j} \tag{A-4}$$

where $X_p = (x_p\cos\theta - y_p\sin\theta + X_g)$

$$Y_p = (x_p\sin\theta + y_p\cos\theta + Y_g) \tag{A-5}$$

Equations (A-5) can be differentiated to yield the relative velocities $\dot{X}_p$ and $\dot{Y}_p$ for the projectile CG as follows.

$$\dot{X}_p = -x_p\sin\theta\,\dot{\theta} + \dot{x}_p\cos\theta - y_p\cos\theta\,\dot{\theta} - \dot{y}_p\sin\theta + \dot{X}_g$$
$$\dot{Y}_p = x_p\cos\theta\,\dot{\theta} + \dot{x}_p\sin\theta - y_p\sin\theta\,\dot{\theta} + \dot{y}_p\cos\theta + \dot{Y}_g \tag{A-6}$$

APPENDIX B

PROJECTILE DISPLACEMENT AT BOURRELET

APPENDIX B

PROJECTILE DISPLACEMENT AT BOURRELET

The radial displacement vector of the projectile centerline in the plane of the bourrelet, measured from the bore centerline is (see Figure 22)

$$\vec{r} = \vec{r}_B - \vec{r}_{B^*}$$

$$= (\vec{r}_{B/P} + \vec{r}_{P/G} + \vec{r}_G) - (\vec{r}_{B^*/G} + \vec{r}_G)$$

$$= \vec{r}_{B/P} + \vec{r}_{P/G} - \vec{r}_{B^*/G}$$

$$= \ell_p \cos\alpha\, \hat{\ell} + \ell_p \sin\alpha\, \hat{m} + x_p \hat{\ell} + y_p \hat{m} - \ell_g \hat{\ell} \qquad \text{(B-1)}$$

where P denotes the CG of the projectile and G denotes the CG of the gun, $\hat{\ell}$ and $\hat{m}$ are unit vectors along and perpendicular to the gun axis, and $\hat{i}$ and $\hat{j}$ are unit vectors along the inertial X and Y axes. Here ℓ_g, which is the instantaneous distance of the bourrelet from the gun CG measured along the gun axis, can be seen to be

$$\ell_g \approx x_p + \ell_p \cos\alpha \qquad \text{(B-2)}$$

Using the above approximation, it can be seen that

$$\vec{r} = A\hat{i} + B\hat{j} \qquad \text{(B-3)}$$

where $A = -(\ell_p \sin\alpha + y_p)\sin\theta$

$$B = (\ell_p \sin\alpha + y_p)\cos\theta \qquad \text{(B-4)}$$

Contact occurs when

$$\left|\vec{r}\right| \geq R_{cl,bc} \qquad \text{(B-5)}$$

where $R_{cl,bc}$ is the radial clearance between the bourrelet and the gun bore. The displacement of the projectile into the bore is then

$$\delta_{bc} = \left|\vec{r}\right| - R_{cl,bc} = \left|y_p + \ell_p \sin\alpha\right| - R_{cl,bc} \qquad \text{(B-6)}$$

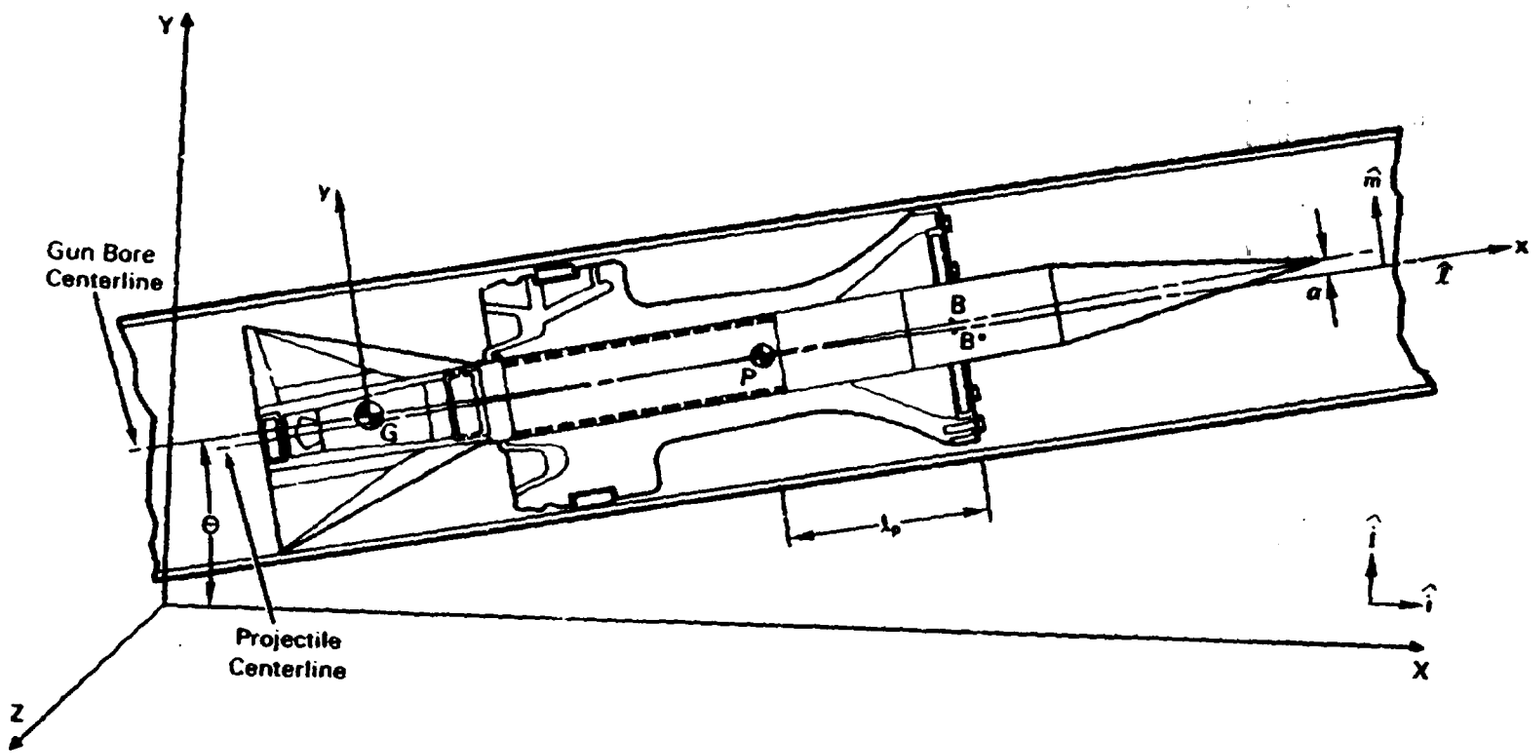

Figure 22. Bourrelet Impacting Bore

APPENDIX C

GENERALIZED FORCES ACTING ON SYSTEM

APPENDIX C

GENERALIZED FORCES ACTING ON SYSTEM

The virtual power developed due to generalized forces through infinitesimal virtual velocities compatible with system constraints can be represented by

$$\delta P_{tot} = \sum_{k=1}^{6} Q_k \cdot \delta v_k = Q_{X_g} \delta \dot{X}_g + Q_{Y_g} \delta \dot{Y}_g + Q_\theta \delta \dot{\theta} + Q_{x_p} \delta \dot{x}_p + Q_{y_p} \delta \dot{y}_p + Q_\alpha \delta \dot{\alpha} \qquad \text{(C-1)}$$

where Q_k represents the nonconservative generated force associated with the generalized velocity v_k in the direction of the k^{th} generalized coordinate. The system virtual power will also be that resulting from the application of given nonconservative forces acting on the system, which is the sum of the following contributions.

δP due to force $f_x(t)$ generated by the gas pressure on the projectile $= f_x(t)\ \delta \dot{x}_p$

δP due to force $f_y(t)$ on projectile $= f_y(t)\ \delta \dot{y}_p$

δP due to moment $m(t)$ on projectile $= m(t)\ \delta \dot{\alpha}$

δP due to friction at contact surface between the bourrelet and the bore

$$= F_{f_{bc}} \delta \dot{x}_p \operatorname{sgn}(\dot{x}_p) = -\mu_{bc} k_{bc} \left(\left| y_p + \ell_p \alpha \right| - R_{cl,bc} \right) \delta \dot{x}_p \operatorname{sgn}(\dot{x}_p)$$

for small α, upon contact.

δP due to quadratic yaw damping $= -(c_1 \dot{\alpha} + c_2 \dot{\alpha} \left| \dot{\alpha} \right|)\ \delta \dot{\alpha}$

Thus, $\delta P_{tot} = \{ f_x(t) - \mu_{bc} k_{bc} \left(\left| y_p + \ell_p \alpha \right| - R_{cl,bc} \right) \operatorname{sgn}(\dot{x}_p) \} \delta \dot{x}_p + \{ f_y(t)\ \delta \dot{y}_p \} + \{ m(t) - (c_1 \dot{\alpha} + c_2 \dot{\alpha} \left| \dot{\alpha} \right|) \} \delta \dot{\alpha} \qquad \text{(C-2)}$

A comparison of equations (C-1) and (C-2) then yields the following generalized forces

$$Q_{x_g} = Q_{Y_g} = Q_\theta = 0$$

$$Q_{x_p} = f_x(t) - \mu_{bc}\, k_{bc} \left(\left| y_p + \ell_p \alpha \right| - R_{cl,bc} \right) \operatorname{sgn} (\dot{x}_p) \qquad \text{(C-3)}$$

$$Q_{y_p} = f_y(t)$$

$$Q_\alpha = m(t) - c_1 \dot{\alpha} - c_2\, \dot{\alpha} \left| \dot{\alpha} \right|$$

In the above, $m(t)$ is any external yawing moment acting on the projectile, μ_{bc} is the coefficient of friction at the bourrelet and bore contact, c_1 and c_2 are first order and second order damping coefficients for the yawing motion of the projectile, and $f_x(t)$ and $f_y(t)$ are applied forces on the projectile in x and y directions given by

$$f_x(t) = p(t)\, A_p \cos \alpha \,, \qquad \text{(C-4)}$$

$$f_y(t) = p(t)\, A_p \sin \alpha \,,$$

where A_p is the area of the projectile cross section on which the gas pressure $p(t)$ acts.

APPENDIX D

COMPUTER PROGRAM LISTING

```
$ucsd$
$sysprog$
program newmark (input, output, keyboard);
(*************************************************************************)
(*                                                                       *)
(*                            N E W M A R K                              *)
(*                      Numerical Integration Program                    *)
(*                                                                       *)
(*   Author: John Baugh, Battelle-Northwest Laboratories                 *)
(*   Date: November 1, 1985                                              *)
(*                                                                       *)
(*   Abstract: This program determines displacements, velocities, and    *)
(*     accelerations using the Newmark numerical integration method for  *)
(*     nonlinear, multi-degree-of-freedom systems.  Both the constant-   *)
(*     average-acceleration and linear acceleration methods are available*)
(*     to the user.  The system matrices (e.g. mass, stiffness, etc.) are*)
(*     defined by subroutines in the file 'USER.TEXT'.  Subroutines are  *)
(*     called at each time step and may be functions of the current      *)
(*     displacement, velocity, and time vectors.                         *)
(*                                                                       *)
(*************************************************************************)

$search 'MOPBOX'$        (* access matrix operations *)
import mopbox;
$search 'PLOTPAC'$       (* access plotting routines *)
import plotpac;

const
  MAXDOFS = 6;
  PI = 3.141592654;
type
  string80 = string[80];
  plot_record =
    record
      displ : plot_array;
      veloc : plot_array;
      accel : plot_array
    end;
var
  outfile : text;
  filename : string80;
  ndofs : integer;
  npts : integer;
  ch : char;
  step_size : real;
  maxtime : real;
  time : plot_array;
  data : array [1..MAXDOFS] of plot_record;

(********************  USER-DEFINED INPUT FUNCTIONS  ********************)

$include 'USER.TEXT'$

(*********************************  UTILITIES  *********************************)

function open_outfile(filename : string80) : boolean;
(* open an output file if possible *)
begin
  try
    rewrite(outfile, filename);
```

```
    open_outfile := true;
  recover
    open_outfile := false
end; (* open_outfile *)


function ctoi (ch : char) : integer;
(* convert a character to an integer digit *)
begin
  ctoi := ord(ch) - ord('0')
end; (* ctoi *)


procedure output_data (npts : integer);
(* output the displs, velocs, and accels to the specified file *)
var
  i, dof : integer;
begin
  writeln(outfile, '  dof      time (mS)        displ        ',
                   'veloc          accel');
  writeln(outfile);
  for i := 1 to npts do
  begin
    for dof := 1 to ndofs do
    begin
      write(outfile, dof:4, '     ');
      write(outfile, time[i]:10:4, '     ');
      write(outfile, data[dof].displ[i]:10:4, '     ');
      write(outfile, data[dof].veloc[i]:10:4, '     ');
      writeln(outfile, data[dof].accel[i]:10:4)
    end;
    writeln(outfile)
  end
end; (* output_data *)


procedure display_plots (dof, npts : integer);
(* display the displ, veloc, and accel plots at the specified dof *)

  procedure display (x, y : plot_array; title : plot_string);
  var
    ch : char;
  begin
    if initgraphics then
    begin
      plot(x, y, npts, 1, title);
      read(keyboard, ch);
      termgraphics
    end
  end; (* display *)

begin
  display(time, data[dof].displ, 'Displacement vs Time');
  display(time, data[dof].veloc, 'Velocity vs Time');
  display(time, data[dof].accel, 'Acceleration vs Time')
end; (* display_plots *)


procedure integrate (method : char; step, maxtime : real; var i : integer);
(*************************************************************************)
```

```
(*                                                                          *)
(*  Abstract: This subroutine carries out the numerical integration         *)
(*    process.  If method = 'l', then the linear acceleration method is     *)
(*    used.  Otherwise, the constant-average-acceleration method is used.   *)
(*    step specifies the time step to be used and maxtime specifies the     *)
(*    length of time over which to integrate.  i returns the total number   *)
(*    of points that were integrated over.  In addition to using the        *)
(*    matrix routines provided by mopbox, integrate calls various user-     *)
(*    defined routines to get information about the particular problem      *)
(*    at hand.  The following routines are called from the file             *)
(*    'USER.TEXT':                                                          *)
(*      get_ndofs - returns the number of degrees of freedom                *)
(*      get_init_values - returns initial displs, velocs, and accels        *)
(*      get_mass - returns the mass matrix                                  *)
(*      get_stiffness - returns the stiffness matrix                        *)
(*      get_damping - returns the damping matrix                            *)
(*      get_force - returns the force vector                                *)
(*                                                                          *)
(****************************************************************************)
type
  int_const = array[0..7] of real;
var
  T : real;
  Dp,Vp,Ap : vector;
  Dc,Vc,Ac : vector;

  mass : matrix;
  damping : matrix;
  stiffness, Keff : matrix;
  F, Feff : vector;
  a : int_const;

  v1,v2,v3,v4,v5 : vector;
  m1,m2 : matrix;

  count : integer;
  npts, nskip : integer;

  procedure save_data(T : real; D, V, A : vector; i : integer);
  (* saves the current data in the global arrays *)
  var
    dof : integer;
  begin
    time[i] := T * 1000.0;
    for dof := 1 to ndofs do
    begin
      data[dof].displ[i] := D[dof];
      data[dof].veloc[i] := V[dof];
      data[dof].accel[i] := A[dof]
    end
  end; (* save_data *)

  procedure get_constants (method : char; step : real; var a : int_const);
  (* returns the integration constants for newmark integration *)
  var
    alpha, delta : real;
  begin
    if (method = 'l') then      (* linear acceleration method *)
      alpha := 1.0 / 6.0
    else                        (* constant average acceleration *)
```

```
      alpha := 0.25;
    delta := 0.5;
    a[0] := 1.0 / (alpha * sqr(step));
    a[1] := delta / (alpha * step);
    a[2] := 1.0 / (alpha * step);
    a[3] := 1.0 / (2.0 * alpha) - 1.0;
    a[4] := delta / alpha - 1.0;
    a[5] := step / 2.0 * (delta / alpha - 2.0);
    a[6] := step * (1.0 - delta);
    a[7] := delta * step
  end; (* get_constants *)

begin (* integrate *)

  ndofs := get_ndofs;            (* get the number of degrees of freedom *)
  if set_size(ndofs) then;       (* set size of matrix operations in mopbox *)
  get_init_values(Dp, Vp, Ap);     (* get the initial values *)
  get_constants(method, step, a);    (* get newmark integration constants *)
  count := 1;                        (* counts each time step taken *)
  i := 0;                            (* the current number of plotting pts *)
  T := 0.0;                          (* start counting at time T = 0 *)

  npts := trunc(maxtime / step) + 1;
  nskip := trunc(npts / MAXPTS) + 1;

  while (T <= maxtime) do
  begin

    if ((count-1) mod nskip) = 0 then
    begin
      i := i + 1;
      save_data(T, Dp, Vp, Ap, i)  (* for output and plotting *)
    end;

    (**** get the mass, damping, stiffness, and force matrices ****)

    get_mass(Dp, Vp, mass);
    get_damping(Dp, Vp, damping);
    get_stiffness(Dp, Vp, stiffness);
    get_force(Dp, Vp, T, F);

    (**** form effective stiffness matrix ****)
    (* Keff := stiffness + a[0]*mass + a[1]*damping *)

    scalem(mass,a[0],m1);
    scalem(damping,a[1],m2);
    addm(m1,m2,m1);
    addm(stiffness,m1,Keff);

    (**** form effective load vector ****)
    (* Feff := F + mass*(a[0]*Dp + a[2]*Vp + a[3]*Ap)
              + damping*(a[1]*Dp + a[4]*Vp + a[5]*Ap) *)

    scalev(Dp,a[0],v1);
    scalev(Vp,a[2],v2);
    scalev(Ap,a[3],v3);
    addv(v1,v2,v1);
    addv(v1,v3,v1);
    mult(mass,v1,v4);
```

```
      scalev(Dp,a[1],v1);
      scalev(Vp,a[4],v2);
      scalev(Ap,a[5],v3);
      addv(v1,v2,v1);
      addv(v1,v3,v1);
      mult(damping,v1,v5);

      addv(F,v4,Feff);
      addv(Feff,v5,Feff);

      (**** solve for displacements ****)
      (* Dc := Keff^-1 * Feff *)

      if not solve(Keff,Feff,Dc) then
        write('can't solve equation system (zero in diagonal)');

      (**** determine accelerations ****)
      (*  Ac := a[0]*(Dc - Dp) - a[2]*Vp - a[3]*Ap *)

      subv(Dc,Dp,v1);
      scalev(v1,a[0],v1);
      scalev(Vp,a[2],v2);
      scalev(Ap,a[3],v3);
      subv(v1,v2,v1);
      subv(v1,v3,Ac);

      (**** determine velocities ****)
      (* Vc := Vp + a[6]*Ap + a[7]*Ac *)

      scalev(Ap,a[6],v1);
      scalev(Ac,a[7],v2);
      addv(Vp,v1,v1);
      addv(v1,v2,Vc);

      (**** reassign current values to previous variables ****)

      copyv(Dc,Dp);       (* Dc -> Dp *)
      copyv(Vc,Vp);
      copyv(Ac,Ap);

      (**** increment time by one time step ****)

      count := count + 1;
      T := (count - 1) * step;
    end

  end; (* integrate *)


  begin (* newmark *)

    writeln('                                   N E W M A R K          ');
    writeln('                              Numerical Integration Program ');
    repeat
      writeln;
      writeln('Solution method --');
      writeln('  1) Linear Acceleration Method');
      writeln('  2) Constant-Average-Acceleration Method');
      write('Enter the number desired : ');
      read(ch);
```

```
    writeln;
  until (ch in ['1','2']);

  write('Enter the step size (mSec): ');
  readln(step_size);
  step_size := 0.001 * step_size;

  write('Enter the length of time (mSec): ');
  readln(maxtime);
  maxtime := 0.001 * maxtime;

  integrate(ch, step_size, maxtime, npts);

  repeat
    write('Output data to what file (<enter> ignores) : ');
    readln(filename);
  until (filename = '') or open_outfile(filename);

  if (filename <> '') then
  begin
    writeln('Writing to file ', filename);
    output_data(npts);
    close(outfile, 'save')
  end;

  writeln('Enter dof number to view (''q'' to quit)');
  repeat
    write('> ');
    read(ch);
    writeln;
    if (ctoi(ch) in [1..ndofs]) then
      display_plots(ctoi(ch), npts);
  until (ch in ['q', 'Q']);

end.  (* newmark *)
```

```
module mopbox;
(*************************************************************************)
(*                                                                       *)
(*                              M O P B O X                              *)
(*                        Matrix OPerations toolBOX                      *)
(*                                                                       *)
(*  Author: John Baugh, Battelle-Northwest Laboratories                  *)
(*  Date: November 1, 1985                                               *)
(*                                                                       *)
(*  Abstract: This module provides low-level operations to manipulate    *)
(*    matrix and vector entities.  In order to use the functions, the    *)
(*    calling program must first call set_size to define the order of    *)
(*    the matrices.  For instance, the call                              *)
(*                          if set_size(8) then ...;                     *)
(*    from the main program results in all matrices being defined as     *)
(*    8x8 and all vectors as 8x1.  It should be noted that attempts      *)
(*    to set the size larger than the maximum declared size (10)         *)
(*    results in set_size returning false (and the size remaining        *)
(*    unchanged).                                                        *)
(*                                                                       *)
(*    The calling program must import this module before it can access   *)
(*    any of its functions, which may be done using the "import" compiler*)
(*    directive.  The module object code must be online during compilation*)
(*    of the calling program (use the "search" compiler directive or     *)
(*    include this module in the system library -- see chapter 1 of the HP*)
(*    Pascal 3.0 Procedure Library for more details).  Before executing  *)
(*    your program, permanently load this module into memory using the   *)
(*    p-load command (see page 120 of the HP Pascal 3.0 User's Guide for *)
(*    more information).                                                 *)
(*                                                                       *)
(*************************************************************************)

export

const
  MAX = 10;       (* maximum size of matrices and vectors *)
type
  vector = array [1..MAX] of real;
  matrix = array [1..MAX, 1..MAX] of real;

function set_size (size : integer) : boolean;
(* set the order for ALL matrix/vector operations in the mopbox module *)

function solve (m1 : matrix; v1 : vector; var v2 : vector) : boolean;
(* solve up to 10 simultaneous linear algebraic equations *)

procedure mult (m1 : matrix; v1 : vector; var v2 : vector);
(* multiply the matrix m1 by vector v1: v2=m1*v1 *)

procedure zerom (var m1 : matrix);
(* zero each element in matrix m1 *)

procedure zerov (var v1 : vector);
(* zero each element in vector v1 *)

procedure copym (m1 : matrix; var m2 : matrix);
(* copy matrix m1 into matrix m2 *)

procedure copyv (v1 : vector; var v2 : vector);
(* copy vector v1 into vector v2 *)
```

```
procedure scalem (m1 : matrix; f1 : real; var m2 : matrix);
(* scale each element in matrix m1 by f1, putting it into matrix m2 *)

procedure scalev (v1 : vector; f1 : real; var v2 : vector);
(* scale each element in vector v1 by f1, putting it into vector v2 *)

procedure addm (m1, m2 : matrix; var m3 : matrix);
(* add matrix m2 to matrix m1: m3=m1+m2 *)

procedure addv (v1, v2 : vector; var v3 : vector);
(* add vector v2 to vector v1: v3=v1+v2 *)

procedure subm (m1, m2 : matrix; var m3 : matrix);
(* subtract matrix m2 from matrix m1: m3=m1-m2 *)

procedure subv (v1, v2 : vector; var v3 : vector);
(* subtract vector v2 from vector v1: v3=v1-v2 *)

implement

var
  n : integer;   (* order of matrix/vector operations *)


function set_size (size : integer) : boolean;
(* set the order for ALL matrix/vector operations in the mopbox module *)
begin
  if (size <= MAX) and (size > 0) then
  begin
    set_size := true;
    n := size;
  end
  else
    set_size := false;
end; (* set_size *)


function solve (m1 : matrix; v1 : vector; var v2 : vector) : boolean;
(*************************************************************************)
(*                                                                       *)
(*   Abstract: This routine solves up to 10 simultaneous linear algebraic *)
(*     equations by Gauss-Jordan Elimination.  If all of the diagonal    *)
(*     terms in the coefficient matrix are non-zero, solve returns true. *)
(*     In general, for an accurate solution, the diagonal terms should   *)
(*     dominate the coefficient matrix, m1.                              *)
(*                                                                       *)
(*     input parameters -                                                *)
(*       m1 : coefficient matrix                                         *)
(*       v1 : right-hand vector                                          *)
(*     output parameters -                                               *)
(*       v2 : solution vector                                            *)
(*       solve : returns true if all terms in the diagonal               *)
(*         of the coefficient matrix are non-zero                        *)
(*                                                                       *)
(*************************************************************************)
label
  1;
var
  i, j, k : integer;
```

```
  hold : real;
begin
  solve := true;
  for i := 1 to n do      (* loop over each of the columns of ml *)
  begin
    for k := 1 to n do     (* loop over each row except pivot row *)
    begin
      if (k <> i) then
      begin
        if (ml[i,i] <> 0.0) then    (* be sure diagonal is non-zero *)
        begin
          hold := -ml[k,i] / ml[i,i];
          for j := 1 to n do        (* loop over each element in a row *)
          begin
            ml[k,j] := ml[k,j] + hold * ml[i,j];
            if (j = i) then
              ml[k,j] := 0.0
          end;
          vl[k] := vl[k] + hold * vl[i]
        end
        else    (* found a zero diagonal term in coefficient matrix, ml *)
        begin
          solve := false;
          goto 1;
        end
      end
    end;
    hold := ml[i,i];
    for j := 1 to n do              (* this loop is for each element in *)
      ml[i,j] := ml[i,j] / hold;    (* the pivot row *)
    ml[i,i] := 1.0;
    vl[i] := vl[i] / hold
  end;
  for i := 1 to n do                (* copy solution into solution into *)
    v2[i] := vl[i];                 (* the solution vector *)
1:
end; (* solve *)


procedure mult (ml : matrix; vl : vector; var v2 : vector);
(* multiply the matrix ml by vector vl: v2=ml*vl *)
var
  i, j : integer;
begin
  for i := 1 to n do
  begin
    v2[i] := 0.0;
    for j := 1 to n do
      v2[i] := v2[i] + ml[i,j] * vl[j]
  end
end; (* mult *)


procedure zerom (var ml : matrix);
(* zero each element in matrix ml *)
var
  i, j : integer;
begin
  for i := 1 to n do
    for j := 1 to n do
```

```
      m1[i,j] := 0.0
end; (* zerom *)


procedure zerov (var v1 : vector);
(* zero each element in vector v1 *)
var
  i : integer;
begin
  for i := 1 to n do
    v1[i] := 0.0
end; (* zerov *)


procedure copym (m1 : matrix; var m2 : matrix);
(* copy matrix m1 into matrix m2 *)
var
  i, j : integer;
begin
  for i := 1 to n do
    for j := 1 to n do
      m2[i,j] := m1[i,j]
end; (* copym *)


procedure copyv (v1 : vector; var v2 : vector);
(* copy vector v1 into vector v2 *)
var
  i : integer;
begin
  for i := 1 to n do
    v2[i] := v1[i]
end; (* copyv *)


procedure scalem (m1 : matrix; f1 : real; var m2 : matrix);
(* scale each element in matrix m1 by f1, putting it into matrix m2 *)
var
  i, j : integer;
begin
  for i := 1 to n do
    for j := 1 to n do
      m2[i,j] := f1 * m1[i,j]
end; (* scalem *)


procedure scalev (v1 : vector; f1 : real; var v2 : vector);
(* scale each element in vector v1 by f1, putting it into vector v2 *)
var
  i : integer;
begin
  for i := 1 to n do
    v2[i] := f1 * v1[i]
end; (* scalev *)


procedure addm (m1, m2 : matrix; var m3 : matrix);
(* add matrix m2 to matrix m1: m3=m1+m2 *)
var
  i, j : integer;
```

```
begin
  for i := 1 to n do
    for j := 1 to n do
      m3[i,j] := m1[i,j] + m2[i,j]
end; (* addm *)


procedure addv (v1, v2 : vector; var v3 : vector);
(* add vector v2 to vector v1: v3=v1+v2 *)
var
  i : integer;
begin
  for i := 1 to n do
    v3[i] := v1[i] + v2[i]
end; (* addv *)


procedure subm (m1, m2 : matrix; var m3 : matrix);
(* subtract matrix m2 from matrix m1: m3=m1-m2 *)
var
  i, j : integer;
begin
  for i := 1 to n do
    for j := 1 to n do
      m3[i,j] := m1[i,j] - m2[i,j]
end; (* subm *)


procedure subv (v1, v2 : vector; var v3 : vector);
(* subtract vector v2 from vector v1: v3=v1-v2 *)
var
  i : integer;
begin
  for i := 1 to n do
    v3[i] := v1[i] - v2[i]
end; (* subv *)

end. (* mopbox *)
```

```
module plotpac;
(*****************************************************************************)
(*                                                                           *)
(*                              P L O T P A C                                *)
(*                         Graphic Plotting Package                          *)
(*                                                                           *)
(*  Author: John Baugh, Battelle-Northwest Laboratories                      *)
(*  Date: August 12, 1985                                                    *)
(*                                                                           *)
(*  Abstract:  This is a library for plotting data in the form of line       *)
(*    graphs.  The only information needed by the plotting routine is        *)
(*    the arrays of points, the number of points, an index for the line      *)
(*    style (e.g. solid=1, dotted=7, etc.), and a title which, when null,    *)
(*    causes the current plot to be displayed over the previous one.         *)
(*    Plots are automatically scaled for convenience.  A typical example     *)
(*    is illustrated below:                                                  *)
(*                                                                           *)
(*              program example (input, output);                             *)
(*              $search 'VOLUME:PLOTPAC'$                                    *)
(*              import plotpac;                                              *)
(*              const                                                        *)
(*                SOLID = 1                                                  *)
(*                DASHED = 2                                                 *)
(*              var                                                          *)
(*                x, y, z : plot_array;                                      *)
(*                keyboard : text;                                           *)
(*                ch : char;                                                 *)
(*              begin                                                        *)
(*                reset(keyboard, 'console:');                               *)
(*                                                                           *)
(*                   < define x, y, and z arrays                             *)
(*                            from 1 to numpts here >                        *)
(*                                                                           *)
(*                if initgraphics then                                       *)
(*                begin                                                      *)
(*                  plot(x, y, numpts, SOLID, 'y and z versus x');           *)
(*                  plot(x, z, numpts, DASHED, '');                          *)
(*                  read(keyboard, ch);                                      *)
(*                  termgraphics                                             *)
(*                end                                                        *)
(*              end.                                                         *)
(*                                                                           *)
(*    As in the example, the calling program must initialize the graphics    *)
(*    display by calling "initgraphics" before plotting data with            *)
(*    procedure "plot".  After plotting the data, a "read" or "readln"       *)
(*    statement located before a call to "termgraphics" keeps the plot       *)
(*    displayed until the user gives an appropriate response.  In this       *)
(*    case, typing any key will remove the plot and return the display to    *)
(*    alpha.                                                                 *)
(*                                                                           *)
(*    The calling program must import this module before it can access       *)
(*    any of its functions, which may be done using the "import" compiler    *)
(*    directive.  The module object code must be online during compilation   *)
(*    of the calling program (use the "search" compiler directive or         *)
(*    include this module in the system library -- see chapter 1 of the HP   *)
(*    Pascal 3.0 Procedure Library for more details).  Before executing      *)
(*    your program, permanently load this module into memory using the       *)
(*    p-load command (see page 120 of the HP Pascal 3.0 User's Guide for     *)
(*    more information).                                                     *)
(*                                                                           *)
```

```
(*****************************************************************************)

import dgl_lib;          (* access graphics library *)

export

const
  MAXPTS = 150;
type
  plot_string = string[80];
  plot_array = array [1..MAXPTS] of real;

function initgraphics : boolean;
(* initialize the device-independent graphics library and display *)

procedure termgraphics;
(* terminate the device-independent graphics library and display *)

procedure plot (var x, y : plot_array; n : integer; linetype : integer;
                title : plot_string);
(* plot the input arrays *)

implement

const
  ASPECT = 3.0;                  (* aspect ratio for all characters *)
type
  modetype = (alpha, graphics);
  roundtype = (up, down, near);
var
  initstate : boolean;           (* true if graphics initialized *)
  xmin, xmax : real;             (* min and max values in x array *)
  ymin, ymax : real;             (* min and max values in y array *)

(****************************** MATH ROUTINES ******************************)

function log (x : real) : real;
(* return log base 10 of x *)
begin
  log := ln(x) / ln(10.0)
end; (* log *)


function power (base, exponent : real) : real;
(* return the value of base ^ exponent *)
begin
  if (base = 0.0) then
    power := 0.0
  else
    if (base > 0.0) then
      power := exp(ln(base) * exponent)
    else
      if (base < 0.0) then
        power := - exp(ln(-base) * exponent)
end; (* power *)


procedure scinotation (x : real; var mantissa, exponent : real);
(* return x in the form of scientific notation *)
```

```
begin
  if (x = 0.0) then
  begin
    mantissa := 0.0;
    exponent := 0.0
  end
  else
    if (x < 0.0) then
    begin
      mantissa := -power(10.0, log(-x) - trunc(log(-x)));
      exponent := trunc(log(-x))
    end
    else
      if (x > 0.0) then
      begin
        mantissa := power(10.0, log(x) - trunc(log(x)));
        exponent := trunc(log(x))
      end;
  if (abs(mantissa) < 1.0) then      (* normalize the result *)
  begin
    mantissa := mantissa * 10.0;
    exponent := exponent - 1.0
  end
end; (* scinotation *)


function absmax (x, y : real) : real;
(* return the maximum of the absolute values of x and y *)
begin
  if (abs(x) > abs(y)) then
    absmax := abs(x)
  else
    absmax := abs(y)
end; (* absmax *)


function round2 (n, m : real; mode : roundtype) : real;
(* round n to the nearest m, according to mode *)
const
  ROUNDOFF = 1E-100;              (* roundoff error fudge factor *)
var
  rounded : real;                 (* temporary holding area *)
  negative : boolean;             (* flag: "is it negative?" *)
begin
  negative := (n < 0.0);          (* is the number negative? *)
  if negative then
  begin
    n := abs(n);                  (* work with a positive number *)
    if mode = up then
      mode := down
    else
      if mode = down then
        mode := up
  end;
  case mode of
    down : rounded := trunc(n / m) * m;
    up   : begin
             rounded := n / m ;
             if abs(rounded - round(rounded)) > ROUNDOFF then
               rounded := (trunc(rounded) + 1.0) * m
```

```
            else
              rounded := trunc(rounded) * m
          end ;
    near : rounded := trunc(n / m + m * 0.5) * m
  end ;
  if negative then                    (* reinstate the sign *)
    rounded := -rounded;
  round2 := rounded
end; (* round2 *)

(******************** MACHINE-DEPENDENT GRAPHICS ROUTINES ********************)

function setmode (mode : modetype) : boolean;
(* set the display mode to alpha or graphics *)
const
  ALPHADISP = 1051;           (* alpha display address *)
  GRAPHDISP = 1050;           (* graphics display address *)
var
  error : integer;            (* output_esc error (0 if ok) *)
  on, off : integer;          (* on/off switches *)
  x : real;                   (* ignored *)
begin
  setmode := true;
  on := 1;
  off := 0;
  if (mode = alpha) then
  begin
    output_esc(GRAPHDISP, 1, 0, off, x, error);  (* graphics off *)
    if (error <> 0) then
      setmode := false;
    output_esc(ALPHADISP, 1, 0, on, x, error);   (* alpha on *)
    if (error <> 0) then
      setmode := false
  end
  else
  begin
    output_esc(ALPHADISP, 1, 0, off, x, error);  (* alpha off *)
    if (error <> 0) then
      setmode := false;
    output_esc(GRAPHDISP, 1, 0, on, x, error);   (* graphics on *)
    if (error <> 0) then
      setmode := false
  end
end; (* setmode *)


function initgraphics : boolean;
(* initialize the device-independent graphics library and display *)
const
  XPIXEL = 400;          (* pixels in x dir (HP9816) *)
  YPIXEL = 300;          (* pixels in y dir (HP9816) *)
  DEVICE = 3;            (* address of screen *)
  CONTROL = 0;           (* ignored by screen *)
var
  error : integer;       (* display init error (0 if ok) *)
begin
  graphics_init;
  display_init(DEVICE, CONTROL, error);
  if (error = 0) and setmode(graphics) then
  begin
```

```
    initstate := true;
    set_aspect(XPIXEL - 1, YPIXEL - 1)     (* use entire screen *)
  end
  else
    initstate := false;
  initgraphics := initstate
end; (* initgraphics *)


procedure termgraphics;
(* terminate the device-independent graphics library and display *)
begin
  if initstate then
  begin
    if setmode(alpha) then
    begin
      graphics_term;              (* terminate graphics device *)
      initstate := false          (* uninitialize graphics *)
    end
  end
end; (* termgraphics *)

(************************** PLOTTING FUNCTIONS **************************)

procedure setcharsize (width : real);
(* set the character size using ASPECT as the aspect ratio *)
begin
  set_char_size(width, ASPECT * width)
end; (* setcharsize *)


procedure get_format (x : real; var numlen, fraclen : integer);
(* determine the required string length (total and fraction) of x *)
begin
  if abs(x) > 0 then
    numlen := trunc(log(abs(x)) + 1)
  else
    numlen := 0;
  fraclen := 4 - numlen;
  if numlen < 1 then
    numlen := 1;
  if fraclen <= 0 then
  begin
    fraclen := 0;
    numlen := numlen + fraclen + 1;
  end
  else
    numlen := numlen + fraclen + 2;
end; (* get_format *)


procedure writenum (x : real; numlen, fraclen : integer);
(* write x on the graphics display at the current location *)
var
  slen : integer;
  s : plot_string;
begin
  strwrite(s, 1, slen, x:numlen:fraclen);
  setstrlen(s, slen - 1);
  gtext(s)
```

```
end; (* writenum *)


procedure writestr (s : plot_string);
(* write string s, centered and at the top of the screen *)
const
  WIDTH = 0.04;
begin
  setcharsize(WIDTH);
  move(-strlen(s) * WIDTH / 2.0, 0.9);
  gtext(s)
end; (* writestr *)


procedure drawbox (x, y : real);
(* draw a box through points (x,y) and (-x,-y) *)
begin
  move( x, y);
  line( x,-y);
  line(-x,-y);
  line(-x, y);
  line( x, y)
end; (* drawbox *)


procedure findmaxmin (var a : plot_array; n : integer; var max, min : real);
(* determine the max and min values of array "a" relative to zero *)
var
  i : integer;
begin
  max := 0.0;
  min := 0.0;
  for i := 1 to n do
  begin
    if a[i] >= max then
      max := a[i];
    if a[i] <= min then
      min := a[i];
  end;
  if (max = 0.0) and (min = 0.0) then
    max := 1.0;
end; (* findmaxmin *)


procedure findspacing(var max, min, spacing : real);
(* determine the spacing and adjust the max and min values appropriately *)
var
  mantissa, exponent : real;     (* mantissa and exponent of absmax(x) *)
  a : integer;                   (* a & b are used to determine spacing *)
  b : real;
begin
  scinotation(absmax(max, min), mantissa, exponent);
  a := trunc(mantissa) + 1;
  b := power(10.0, exponent);
  case a of
    1          : spacing := 0.20 * b;
    2          : spacing := 0.25 * b;
    3,4        : spacing := 0.50 * b;
    5,6,7,8    : spacing := 1.00 * b;
    9,10       : spacing := 2.00 * b
```

```
    end;
    max := round2(max, spacing, up);
    min := round2(min, spacing, down)
  end; (* findspacing *)


  function glx (xglobal : real) : real;
  (* convert global x coordinates to local x coordinates *)
  begin
    glx := 2.0 * (xglobal - xmin) / (xmax - xmin) - 1.0
  end; (* glx *)


  function gly (yglobal : real) : real;
  (* convert global y coordinates to local y coordinates *)
  begin
    gly := 2.0 * (yglobal - ymin) / (ymax - ymin) - 1.0
  end; (* gly *)


  procedure drawaxes;
  (* draw the x and y axes *)
  begin
    move(glx(xmin), gly(0.0));
    line(glx(xmax), gly(0.0));
    move(glx(0.0), gly(ymin));
    line(glx(0.0), gly(ymax))
  end; (* drawaxes *)


  function in_range(x, y : real) : boolean;
  (* determine whether or not point (x,y) lies inside the display area *)
  begin
    if (x <= xmax) and (x >= xmin) and (y <= ymax) and (y >= ymin) then
      in_range := true
    else
      in_range := false
  end; (* in_range *)


  procedure drawlines (var x, y : plot_array; n : integer);
  (* draw lines connecting the points in the x and y arrays *)
  var
    i : integer;
    count : integer;
  begin
    count := 1;
    while (not in_range(x[count], y[count])) and (count < n) do
      count := count + 1;
    move(glx(x[count]), gly(y[count]));
    if count < n then
      for i := (count + 1) to n do
        if in_range(x[i], y[i]) then
          line(glx(x[i]), gly(y[i]))
  end; (* drawlines *)


  procedure labelxaxis (xspacing : real);
  (* put tick marks and scale numbers on the x axis *)
  const
```

```
  WIDTH = 0.03;          (* width of a character (local) *)
  SEMI = 0.02;           (* half length of a major tick mark *)
var
  xg : real;                      (* global x coordinate *)
  xl, yl : real;                  (* local x and y coordinates *)
  numlen, fraclen : integer;      (* number and fraction lengths *)
  height : real;                  (* character height *)
begin
  setcharsize(WIDTH);
  height := ASPECT * WIDTH;
  get_format(absmax(xmax, xmin), numlen, fraclen);
  yl := gly(0.0);
  xg := xmin;

  while (xg <= (xmax + 0.5 * xspacing)) do
  begin
    if abs(xg) < (0.5 * xspacing) then          (* x is zero *)
      if (ymin < 0.0) and (ymax > 0.0) then     (* y axis crosses x axis *)
        xg := xg + xspacing;                    (* skip to the next label *)
    xl := glx(xg);                      (* convert x from global to local *)
    move(xl, yl - SEMI);                (* move to the proper location *)
    line(xl, yl + SEMI);                (* and draw the tick mark *)

    if (ymax <= 0.0)                    (* x axis at top of the screen *)
      move(xl - numlen * WIDTH / 2.0, yl + 0.5 * height)
    else                                (* x axis is somewhere below top *)
      move(xl - numlen * WIDTH / 2.0, yl - 1.5 * height);

    writenum(xg, numlen, fraclen);
    xg := xg + xspacing
  end
end; (* labelxaxis *)


procedure labelyaxis (yspacing : real);
(* put tick marks and scale numbers on the y axis *)
const
  WIDTH = 0.03;          (* width of a character (local) *)
  SEMI = 0.02;           (* half length of a major tick mark *)
var
  yg : real;                      (* global y coordinate *)
  xl, yl : real;                  (* local x and y coordinates *)
  numlen, fraclen : integer;      (* number and fraction lengths *)
  height : real;                  (* character height *)
begin
  setcharsize(WIDTH);
  height := ASPECT * WIDTH;
  get_format(absmax(ymax, ymin), numlen, fraclen);
  xl := glx(0.0);
  yg := ymin;

  while (yg <= (ymax + 0.5 * yspacing)) do
  begin
    if abs(yg) < (0.5 * yspacing) then          (* y is zero *)
      if (xmin < 0.0) and (xmax > 0.0) then     (* x axis crosses y axis *)
        yg := yg + yspacing;                    (* skip to the next label *)
    yl := gly(yg);                      (* convert y from global to local *)
    move(xl - SEMI, yl);                (* move to the proper location *)
    line(xl + SEMI, yl);                (* and draw the tick mark *)
```

```
    if (xmax <= 0.0) then           (* y axis is at the right edge *)
      move(x1 + WIDTH, y1 - 0.25 * height)
    else                            (* y axis somewhere left of it *)
      move(x1 - WIDTH * (numlen + 2), y1 - 0.25 * height);

    writenum(yg, numlen, fraclen);
    yg := yg + yspacing
  end
end; (* labelyaxis *)


procedure plot (var x, y : plot_array; n : integer; linetype : integer;
                title : plot_string);
{*************************************************************************}
{*                                                                       *}
{*  Abstract: This routine plots the input arrays as a line graph.  The  *}
{*    x array should contain the horizontal coordinates and the y array  *}
{*    should contain vertical coordinates.  n is the total number of     *}
{*    points, which must be greater than zero.  linetype is an integer   *}
{*    specifying the desired line style -                                *}
{*       1 - solid line                                                  *}
{*       2 - dashed line                                                 *}
{*       7 - dotted line                                                 *}
{*    (see page 325 of the HP Pascal 3.0 Graphics Techniques Manual for  *}
{*    more styles).  title specifies the title for the plot, unless it   *}
{*    is declared a null string, in which case the current plot is       *}
{*    displayed over the previous one.                                   *}
{*                                                                       *}
{*************************************************************************}
var
  xspacing, yspacing : real;        (* spacing for tick marks *)
begin
  if initstate and (n > 0) then
  begin
    if (title <> '') then      (* a new plot *)
    begin
      set_line_style(1);
      writestr(title);
      drawbox(1, 1);
      set_viewport(0.12, 0.88, 0.07, 0.65);
      findmaxmin(x, n, xmax, xmin);
      findmaxmin(y, n, ymax, ymin);
      findspacing(xmax, xmin, xspacing);
      findspacing(ymax, ymin, yspacing);
      drawaxes;
      labelxaxis(xspacing);
      labelyaxis(yspacing)
    end;
    set_line_style(linetype);
    drawlines(x, y, n)
  end
end; (* plot *)

end. (* plotpac *)
```

```
(*****************************************************************************)
(*                                                                           *)
(*                 USER DATA FOR NEWMARK INTEGRATION PROGRAM                 *)
(*                    (SIX-DOF IN-BORE PROJECTILE MODEL)                     *)
(*                                                                           *)
(*****************************************************************************)

(*********************  GLOBAL VARIABLE DECLARATIONS  ********************)

  g    : real;        { gravitational acceleration constant }
  mp   : real;        { mass of projectile }
  mg   : real;        { mass of gun }
  Ip   : real;        { mass moment of inertia of projectile }
  Ig   : real;        { mass moment of inertia of gun }
  R    : real;        { radial clearance between gun barrel and projectile }
  mu   : real;        { coefficient of friction at bourrelet (zero) }
  Lp   : real;        { distance between obturator and bourrelet }
  Lb   : real;        { distance between projectile c.g. and bourrelet }
  Lo   : real;        { distance between projectile c.g. and obturator }
  Ka   : real;        { angular stiffness at the obturator }
  Kb   : real;        { bourrelet stiffness (linear) }
  Ko   : real;        { nylon obturator stiffness (linear) }
  Ko1  : real;        { aluminum stiffness at obturator }
  C1   : real;        { angular damping coefficient }
  C2   : real;        { nonlinear angular damping coefficient }
  P    : real;        { maximum applied pressure to projectile }
  area : real;        { cross-sectional area of projectile }

(*****************************  INPUT FUNCTIONS  *****************************)

function get_ndofs : integer;
(* set the number of dofs *)
begin
  get_ndofs := 8
end; (* get_ndofs *)


procedure get_init_values (var displ, veloc, accel : vector);
(* specify the initial values *)
var
  yo : real;
begin
  zerov(displ);
  zerov(veloc);
  zerov(accel);

  g := 386.4;         { in/sec^2 }
  mp := 16.0 / g;     { lb-sec^2/in }
  mg := 4000.0 / g;   { lb-sec^2/in }
  Ip := 0.583;        { lb-in-sec^2 }
  Ig := 150.0;        { lb-in-sec^2 }
  R := 0.0052;        { in }
  Lp := 5.5;          { in }
  Ka := 8.6E5;        { in-lb/rad }
  Ko := 5.0E6;        { lb/in }
  C1 := 71.0;         { lb-in-sec (about 5 percent of critical) }
  C2 := 0.0;
  P := 50.0E3;        { lb/in^2 }
  area := 17.53;      { in^2 }
```

```
  write('Enter the distance from the obturator to the c.g. (in.) : ');
  readln(Lo);
  write('Enter the initial projectile angle (radians) : ');
  readln(displ[6]);
  write('Enter the initial vertical displacement at the obturator (in.) : ');
  readln(yo);
  Lb := Lp - Lo;
  displ[5] := yo + Lo * sin(displ[6]);
end; (* get_init_values *)


procedure get_mass (displ, veloc : vector; var mass : matrix);
(* specify the mass matrix *)
var
  Xg, Xgdot : real;
  Yg, Ygdot : real;
  O, Odot : real;
  xp, xpdot : real;
  yp, ypdot : real;
  a, adot : real;
begin
  Xg := displ[1];    Xgdot := veloc[1];
  Yg := displ[2];    Ygdot := veloc[2];
  O := displ[3];     Odot := veloc[3];
  xp := displ[4];    xpdot := veloc[4];
  yp := displ[5];    ypdot := veloc[5];
  a := displ[6];     adot := veloc[6];

  zerom(mass);

  mass[1,1] := mg + mp;
  mass[1,3] := -mp * (xp*sin(O) + yp*cos(O));
  mass[1,4] := mp * cos(O);
  mass[1,5] := -mp * sin(O);

  mass[2,2] := mg + mp;
  mass[2,3] := mp * (xp*cos(O) - yp*sin(O));
  mass[2,4] := mp * sin(O);
  mass[2,5] := mp * cos(O);

  mass[3,1] := -mp * (xp*sin(O) + yp*cos(O));
  mass[3,2] := mp * (xp*cos(O) - yp*sin(O));
  mass[3,3] := Ig + mp * (sqr(xp) + sqr(yp));
  mass[3,4] := -mp * yp;
  mass[3,5] := mp * xp;

  mass[4,1] := mp;
  mass[4,2] := mp;
  mass[4,3] := mp * (xp*(cos(O) - sin(O)) - yp*(sin(O) + cos(O)));
  mass[4,4] := mp * (sin(O) + cos(O));
  mass[4,5] := mp * (cos(O) - sin(O));

  mass[5,1] := -mp * sin(O);
  mass[5,2] := mp * cos(O);
  mass[5,3] := mp * xp;
  mass[5,5] := mp;

  mass[6,6] := Ip
end; (* get_mass *)
```

```
procedure get_damping (displ, veloc : vector; var damping : matrix);
(* specify the damping matrix *)
var
  Xg, Xgdot : real;
  Yg, Ygdot : real;
  O, Odot : real;
  xp, xpdot : real;
  yp, ypdot : real;
  a, adot : real;
begin
  Xg := displ[1];     Xgdot := veloc[1];
  Yg := displ[2];     Ygdot := veloc[2];
  O := displ[3];      Odot := veloc[3];
  xp := displ[4];     xpdot := veloc[4];
  yp := displ[5];     ypdot := veloc[5];
  a := displ[6];      adot := veloc[6];

  zerom(damping);

  damping[1,4] := -2.0 * mp * sin(O) * Odot;
  damping[1,5] := -2.0 * mp * cos(O) * Odot;

  damping[2,3] := 2.0 * mp * (xpdot*cos(O) - ypdot*sin(O));

  damping[3,1] := -mp * ((xpdot*sin(O) + ypdot*cos(O))
                  + Odot*(xp*cos(O) - yp*sin(O)));
  damping[3,2] := mp * ((xpdot*cos(O) - ypdot*sin(O))
                  - Odot*(xp*sin(O) + yp*cos(O)));
  damping[3,3] := -mp * ((Xgdot*yp - Ygdot*xp) * sin(O)
                  - (Xgdot*xp + Ygdot*yp) * cos(O));
  damping[3,4] := mp * (Xgdot*sin(O) - Ygdot*cos(O));
  damping[3,5] := mp * (Ygdot*sin(O) + Xgdot*cos(O));

  damping[4,1] := mp * sin(O) * Odot;
  damping[4,2] := -mp * cos(O) * Odot;
  damping[4,3] := -mp * (xp*Odot + ypdot);
  damping[4,4] := 2.0 * mp * Odot * (cos(O) - sin(O));
  damping[4,5] := -2.0 * mp * Odot * (sin(O) + cos(O));

  damping[5,1] := -mp * cos(O) * Odot;
  damping[5,2] := -mp * sin(O) * Odot;
  damping[5,3] := mp * (Xgdot*cos(O) + Ygdot*sin(O));
  damping[5,4] := 2.0 * mp * Odot;

  damping[6,6] := C1 + C2 * abs(adot)
end; (* get_damping *)


procedure get_stiffness (displ, veloc : vector; var stiffness : matrix);
(* specify the stiffness matrix *)
var
  Xg, Xgdot : real;
  Yg, Ygdot : real;
  O, Odot : real;
  xp, xpdot : real;
  yp, ypdot : real;
  a, adot : real;
  Ro, Rb : real;
begin
```

```
  Xg := displ[1];     Xgdot := veloc[1];
  Yg := displ[2];     Ygdot := veloc[2];
  O := displ[3];      Odot := veloc[3];
  xp := displ[4];     xpdot := veloc[4];
  yp := displ[5];     ypdot := veloc[5];
  a := displ[6];      adot := veloc[6];

  zerom(stiffness);

  Ro := yp - Lo * sin(a);
  Rb := yp + Lb * sin(a);

  if (abs(Ro) >= R) then      (* contact at obturator *)
    Kol := 1.0E8
  else
    Kol := 0.0;

  if (abs(Rb) >= R) then      (* contact at bourrelet *)
  begin
    Kb := 5.0E7;
    mu := 0.0
  end
  else
  begin
    Kb := 0.0;
    mu := 0.0
  end;

  stiffness[1,4] := -mp * cos(O) * sqr(Odot);
  stiffness[1,5] := mp * sin(O) * sqr(Odot);

  stiffness[2,4] := -mp * sin(O) * sqr(Odot);
  stiffness[2,5] := -mp * cos(O) * sqr(Odot);

  stiffness[3,4] := 2.0 * mp * Odot * xpdot;
  stiffness[3,5] := 2.0 * mp * Odot * ypdot;

  stiffness[4,4] := -mp * sqr(Odot) * (cos(O) + sin(O));
  stiffness[4,5] := mp * sqr(Odot) * (sin(O) - cos(O));

  stiffness[5,5] := -mp * sqr(Odot) + Kb + Kc + Kol;
  stiffness[5,6] := Kb * Lb - (Ko + Kol) * Lo;

  stiffness[6,5] := Kb * Lb - (Ko + Kol) * Lo;
  stiffness[6,6] := Kb * sqr(Lb) + (Ko + Kol) * sqr(Lo) + Ka
end; (* get_stiffness *)


procedure get_force (displ, veloc : vector; time : real; var force : vector);
(* specify the force vector *)
var
  Xg, Xgdot : real;
  Yg, Ygdot : real;
  O, Odot : real;
  xp, xpdot : real;
  yp, ypdot : real;
  a, adot : real;
  load, F : real;

  function sgn(x : real) : real;
```

```
    { return the sign of x }
  begin
    if x < 0.0 then
      sgn := -1.0
    else
      sgn := 1.0
  end; { sgn }

begin
  Xg := displ[1];     Xgdot := veloc[1];
  Yg := displ[2];     Ygdot := veloc[2];
  O := displ[3];      Odot := veloc[3];
  xp := displ[4];     xpdot := veloc[4];
  yp := displ[5];     ypdot := veloc[5];
  a := displ[6];      adot := veloc[6];

  zerov(force);

  if (time < 0.0025) then
    load := - 0.1 + 500.0 * time
  else
    load := 1.5 - 180.0 * time;

  if (load > 1.0) then
    load := 1.0
  else
    if (load < 0.0) then
      load := 0.0;

  F := load * P * area;

  force[2] := -g * (mp + mg);
  force[4] := -mp * g * sin(O) - mu * Kb * (abs(yp + Lb*a) - R) * sgn(xpdot)
              + F * cos(a);
  force[5] := Kb * R * sgn(yp + Lb * a) - mp * g * cos(O) + F * sin(a)
              + R * Kol * sgn(yp - Lo * a);
  force[6] := Kb * R * Lb * sgn(yp + Lb * a)
              - R * Kol * Lo * sgn(yp - Lo * a);
end; { get_force }
```

```
{*****************************************************************************}
{*                                                                           *}
{*                  USER DATA FOR NEWMARK INTEGRATION PROGRAM                *}
{*                                                                           *}
{*****************************************************************************}

{*****************************  INPUT FUNCTIONS  *****************************}

function get_ndofs : integer;
(* set the number of dofs *)
begin
  get_ndofs := 2
end; (* get_ndofs *)


procedure get_init_values (var displ, veloc, accel : vector);
(* specify the initial values *)
begin
  zerov(displ);
  zerov(veloc);
  zerov(accel);
  accel[2] := 10.0
end; (* get_init_values *)


procedure get_mass (displ, veloc : vector; var mass : matrix);
(* specify the mass matrix *)
begin
  zerom(mass);
  mass[1,1] := 2.0;
  mass[2,2] := 1.0
end; (* get_mass *)


procedure get_damping (displ, veloc : vector; var damping : matrix);
(* specify the damping matrix *)
begin
  zerom(damping)
end; (* get_damping *)


procedure get_stiffness (displ, veloc : vector; var stiffness : matrix);
(* specify the stiffness matrix *)
begin
  zerom(stiffness);
  stiffness[1,1] := 6.0;   stiffness[1,2] :=-2.0;
  stiffness[2,1] :=-2.0;   stiffness[2,2] := 4.0
end; (* get_stiffness *)


procedure get_force (displ, veloc : vector; time : real; var force : vector);
(* specify the force vector *)
begin
  zerov(force);
  force[2] := 10.0
end; (* get_force *)
```

LIST OF SYMBOLS

a_n	coefficient of n^{th} term in power series representation of the nonlinear foundation moment
A_p	area of projectile cross section on which gas pressure acts
c_1, c_2	first and second order damping coefficients of projectile resistance to yawing motion
[C]	damping matrix in system equations of motion
f_x, f_y	applied forces on projectile in x and y directions
$F_{f_{bc}}$	friction force on projectile at contact between bourrelet and gun-bore
{F}	column vector of forces in system equations of motion
g	acceleration due to gravity; also denotes gun
G	denotes gun CG
i, j	unit vectors in X, Y directions
I_g	mass moment of inertia of gun about an axis through its CG perpendicular to plane of motion
I_p	mass moment of inertia of projectile about an axis through its CG perpendicular to plane to motion
k_{bc}	spring constant for bourrelet/gun-bore contact spring deflection model
k_o	stiffness of obturator plastic band in obturator transverse spring model
k_o'	stiffness of metallic part in obturator transverse spring model
[K]	stiffness matrix in system equations of motion
ℓ_g	instantaneous distance of bourrelet from gun CG measured along gun bore axis
ℓ_p	distance between projectile CG and plane of bourrelet
ℓ_o	distance between projectile CG and plane of obturator
ℓ, m	unit vectors along x, y directions

L	Lagrangian function
m_g	mass of gun
m_p	mass of projectile
$m(t)$	applied yawing moment on projectile
$[M]$	inertia matrix in system equations of motion
n	number of terms in power series used to represent the nonlinear foundation moment
p	denotes projectile
$p(t)$	time history of gas pressure on projectile
P	denotes projectile CG
δP	virtual power
q_r	r^{th} generalized coordinate
$\{q\}$	column vector of generalized coordinates in system equations of motion
$Q_{r_{nc}}$	nonconservative generalized force associated with r^{th} generalized coordinate q_r
$\vec{r}$	radial displacement vector of projectile centerline in plane of bourrelet measured from the centerline of the gun bore
$R_{cl,\,bc}$	radial clearance between bourrelet and gun bore
$R_{cl,o}$	radial clearance between obturator and gun bore
t	time
T	kinetic energy
v_k	generalized velocity in direction of k^{th} generalized coordinate
V	potential energy
x, y, z	gun body-fixed axes of coordinates
x_p, y_p	coordinates of projectile CG relative to gun body-fixed axes of coordinates
X, Y, Z	inertial frame of reference

X_g, Y_g	displacement of gun CG relative to inertial frame of reference along X, Y axes
α	angular (yawing) displacement of projectile
δ_{bc}	displacement of projectile into the gun bore at contact between bourrelet and gun bore
δ_o	projectile displacement at obturator
θ	angular displacement of gun in direction of positive Z axis
μ_{bc}	coefficient of friction for bourrelet/gun-bore contact

DISTRIBUTION LIST

No. of Copies	Organization
1	Director of Defense Research & Engineering (OSD) ATTN: J. Persh Washington, DC 20301
1	Director Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209
1	Director Institute of Defense Analysis ATTN: Documents Acquisition 1801 Beauregard Street Alexandria, VA 22311
1	HQDA (DAMA-MS) Washington, DC 20310
1	HQDA (DAMA-ZA) Washington, DC 20310
1	HQDA (DAMA-ZD) Washington, DC 20310
1	HQDA (DAMA-ARZ-A) Washington, DC 20310
2	HQDA (DAMA-CSM-VA) (DAMA-CSM-CA) Washington, DC 20310
1	HQDA (DAMA-CSS-T) Washington, DC 20310
10	Central Intelligence Agency Office of Central Reference Dissemination Branch Room GE-47 HQS Washington, DC 20502
1	Commander US Army Materiel Command ATTN: DRCDMD-ST 5001 Eisenhower Avenue Alexandria, VA 22333
1	Commander US Army Materiel Command ATTN: DRCLDC 5001 Eisenhower Avenue Alexandria, VA 22333
1	Commander US Army Materiel Command ATTN: DRCDE 5001 Eisenhower Avenue Alexandria, VA 22333
1	Commander US Army Materiel Command ATTN: DRCDE-R 5001 Eisenhower [illegible]e Alexandria, VA 2[illegible]
1	Commander US Army Materiel Command 5001 Eisenhower Avenue Alexandria, VA 22333
4	Project Manager Cannon Artillery Weapons Systems, PM-CAWS ATTN: AMCPM-CWA-S (R. DeKleine) Picatinny Arsenal, NJ 07801-5001
4	Project Manager Tank Main Armament Systems, PM-TMAS ATTN: AMCPM-TMA Picatinny Arsenal, NJ 07801-5001

DISTRIBUTION LIST

No. of Copies	Organization
14	Commander, USA ARDEC ATTN: SMCAR-CCH (R. Sayer) SMCAR-CCH (S. Slota) SMCAR-CCH (S. Musalli) SMCAR-CCH (J. Delorenzo) SMCAR-CCH (E. Fennel) SMCAR-CCH (B. Konrad) SMCAR-CCH (R. Price) SMCAR-CCH (L. Rosendorf) SMCAR-FSA (T. Davidson) SMCAR-FSA-IM (Botticelli) SMCAR-FSA-IM (W. Smith) SMCAR-SCL-CA (R. Trifiletti) SMCAR-SCL-CA (E. Malatesta) SMCAR-SCL-CA (C. Miller) Picatinny Arsenal, NJ 07801-5001
3	Battelle Pacific Northwest Lab ATTN: Mr. Mark Smith (2 copies) Mr. Mark Garnich P.O. Box 999 Richland, WA 99352
1	Commanding Officer Naval Weapons Support Center ATTN: CODE2024, J. Barber Crane, IN 47522-5020
1	AAI Corporation ATTN: J. Hebert P.O. Box 6767 Baltimore, MD 21204
1	Aerojet Ordnance Corporation ATTN: E. Danials 2521 Michelle Drive Tustin, CA 92680-7014
1	Chamberlain Manufacturing Company ATTN: T. Lynch 550 Ester Street P.O. Box 2335 Waterloo, Iowa 50704
1	Ford Aerospace & Communications International, Inc. ATTN: C. White Ford Road Newport Beach, CA 92658
1	Commander US Army ARRCOM ATTN: SARRI-RLS Rock Island, IL 61299
1	General Defense Corporation Flinchbaugh Division ATTN: Mr. Macelroy 200 E. High St., P.O. Box 127 Red Lion, PA 12356
1	Commander USA TECOM ATTN: AMSTE-TA-R (L. Sabier) APG, MD 21005-5055
26	Director U.S. Army Ballistic Research Lab ATTN: SLCBR-IB-M (B. Burns 5 cpys) SLCBR-IB-M (W. Drysdale) SLCBR-IB-M (K. Bannister) SLCBR-IB-M (L. Burton) SLCBR-IB-M (R. Kaste SLCBR-IB-M (D. Hopkins) SLCBR-IB-M (J. Bender) SLCBR-IB-M (W. Donovan) SLCBR-IB-M (T. Erline) SLCBR-IB-M (E. Patton) SLCBR-IB-M (R. Murray) SLCBR-IB-M (R. Kirkendall) SLCBR-IB-A (A. Horst) SLCBR-IB-A (T. Minor) SLCBR-IB-A (D. Kruczynski) SLCBR-IB-I (A. Barrows) SLCBR-IB (I. May) SLCBR-IB-B (R. Morrison) SLCBR-IB-P (J. Rocchio) SLCBR-TB (W. Kitchens) SLCBR-TB-A (W. Bruchey) SLCBR-D APG, MD 2L005-5066
3	Honeywell, Inc. Defense Systems Division ATTN: C. Candland K. Sundeen G. Campbell 7225 Northland Drive Brooklyn Park, MN 55428

DISTRIBUTION LIST

No. of Copies	Organization
2	Olin Corporation Winchester Group ATTN: D. Marlow H. Perkinson 707 Berkshire Street East Alton, IL 62024-1174
6	Director, USA ARDEC Benet Weapons Laboratory ATTN: SMCAR-LCB-RA T. Simkins G. Pflegl R. Racicot SMCAR-LCB-D J. Zweig SMCAR-LCB-DS J. Santini SMCAR-LCB-M J. Purtell Watervliet, NY 12189
2	Commander U.S. Army AMCCOM, ARDEC Product Assurance Directorate ATTN: SMCAR-QA Picatinny Arsenal, NJ 07801
1	Commander U.S. Army Aviation Research and Development Command ATTN: AMSAV-E 4300 Goodfellow Blvd St. Louis, MO 63120
2	Director US Army Air Mobility Research and Development Laboratory ATTN: Dr. Hans Mark Dr. R.L. Cohen Ames Research Center Moffett Field, CA 94035
2	Director USA Research & Technology Lab Ames Research Center Moffett Field, CA 94035
2	Director USA Research & Technology Lab ATTN: DAVDL-AS Ames Research Center Moffett Field, CA 94035
3	Commander US Army Harry Diamond Labs ATTN: SLCHD-I-TR, H. Curchak SLCHD-I-TR, H. Davis SLCHD-TA-L SLCHD-S-QE-ES, B. Banner 2800 Powder Mill Road Adelphi, MD 20783-1145
1	Commander US Army Harry Diamond Labs ATTN: SLCHD-TA-L 2800 Powder Mill Road Adelphi, MD 20783
1	Commander US Army Missile Command Research, Development & Engr Ctr ATTN: AMSMI-RD AMSMI-YDL Redstone Arsenal, AL 35898-5500
2	Commandant US Army Infantry School ATTN: ATSH-CD-CSO-OR Fort Benning, GA 31905
2	Commander US Army Missile Command ATTN: DRCPM-TO DRCPM-HD Redstone Arsenal, AL 35898
1	Commander US Army Mobility Equipment Research & Dev Command Fort Belvoir, VA 22060
2	Commander US Army Tank Automotive Cmd ATTN: DRSTA-TSL DRSTA-ZSA, R. Beck DRSTA-NS Warren, MI 48090

DISTRIBUTION LIST

No. of Copies	Organization
1	Commander US Army Natick Research and Development Command ATTN: DRDNA-DT Natick, MA 01762
1	Director US Army TRADOC Systems Analy Activity ATTN: ATAA-SL White Sands Missile Range WSMR, NM 88002
2	President US Army Armor and Engr Board ATTN: ATZK-AE-CV ATZK-AE-IN, L. Smith Fort Knox, KY 40121
2	Commander US Army Research Office ATTN: R. Singleton J. Wu J. Chandra P.O. Box 12211 Research Triangle Park, NC 27709-2211
1	Commander US Army Research Office ATTN: Technical Director Engineering Division Metallurgy & Materials Div P.O. Box 12211 Research Triangle Park, NC 27709
1	Director US Army Materials Technology Lab ATTN: Director (5 cys) Watertown, MA 02172
3	Commander US Army Materials and Mechanics Research Center ATTN: J. Mescall Tech Library B. Halpin Watertown, MA 02172
1	Commander US Army Training and Doctrine Command ATTN: TRADOC Library, Fort Monroe, VA 23651
1	Commander US Army Armor School Fort Knox, KY 40121
1	Commander US Army Field Artillery Schl ATTN: Field Artillery Agency Fort Sill, OK 73503
1	Superintendent Naval Postgraduate School ATTN: Director of Library Monterey, CA 93940
1	Commander U.S. Army Combined Arms Combat Development Activity Ft Leavenworth, KS 66027-5080
1	Commander US Army Combat Development Experimentation Command ATTN: Tech Info Center Bldg. 2925, Box 22 Fort Ord, CA 93941
1	Commander Naval Sea Systems Command ATTN: 9132 Washington, DC 20362
1	Commander Naval Sea Systems Command ATTN: SEA-62R41, L. Pasiuk Washington, DC 20362
1	Commander Naval Research Laboratory Development Center ATTN: David W. Taylor Bethesda, MD 20084

DISTRIBUTION LIST

No. of Copies	Organization
5	Commander, Naval Research Lab ATTN: W. Ferguson C. Sanday H. Pusey Tech Library Washington, DC 20375
3	Commander Naval Surface Weapons Center ATTN:Code X211, Library Silver Spring, MD 20910
3	Commander Naval Surface Weapons Center ATTN: Code E-31, R.C. Reed M. Walchak Code V-14, W. Hinckley Silver Spring, MD 20910
5	Commander Naval Surface Weapons Center ATTN: Code G-33, T.N. Tschirn Code n-43, J.J. Yagla L. Anderson G. Soo Hoo Code TX, W.G. Soper Dahlgren, VA 22448
1	Commander Naval Weapons Center China Lake, CA 93555
2	Commander U.S. Naval Weapons Center ATTN: Code 3835 Code 3431, Tech Lib China Lake, CA 93555
2	Commander U.S Naval Ordnance Station Indian Head, MD 20640
2	Commander U.S. Naval Weapons Center ATTN: Code 608 Code 4505 China Lake, CA 93555
3	Commander U.S. Naval Weapons Center ATTN: Code 4057 Code 3835, Code 3835, China Lake, CA 93555
1	Commandant US Marine Corps ATTN: AX Washington, DC 20380
1	AFATL/DLXP ATTN: W. Dittrich; DLJM Eglin AFB, FL 32542
1	AFATL/DLD ATTN: D. Davis Eglin AFB, FL 32542
1	ADTC/DLODL, Tech Lib Eglin AFB, FL 32542
1	AFWL/SUL Kirtland AFB, NM 87117
1	AFWL/SUL ATTN: J.L. Bratton Kirtland AFB, NM 87117
1	AFML (Dr. T. Nicholas) Wright-Patterson AFB, OH 45433
2	ADS (XROT, G. Bennett; ENSSS, Martin Lentz) Wright-Patterson AFB, OH 45433
6	Battelle Pacific Northwest Laboratory ATTN: M.T. Smith M.R. Garnich F.A. Simonen C. Lavender K.A. Ansari J.W. Baugh, Jr. P.O. Box 999 Richland, WA 99352

DISTRIBUTION LIST

No. of Copies	Organization
1	Bell Telephone Labs, Inc. Mountain Avenue Murray Hill, NJ 07971
1	Director Lawrence Livermore Laboratory P.O. Box 808 Livermore, CA 94550
2	Director Lawrence Livermore Laboratory ATTN:E. Farley, L9 D. Burton, L200 Livermore, CA 94550
5	Director Lawrence Livermore Laboratory ATTN: J. Lepper F. Magness R.D. Christensen M.L. Wilkins R. Werne Livermore, CA 94550
1	Director NASA - Ames Research Center Moffett Field, CA 94035
2	Forrestal Research Center Aeronautical Engineering Lab ATTN: S. Lam A. Eringen Princeton, NJ 08540
5	Sandia National Laboratories ATTN: G. Benedetti W. Robinson K. Perano P. Neilan D. Dawson Livermore, CA 94550
1	AFFL.M, The Rand Corporation ATTN: Library-D 1700 Main Street Santa Monica, CA 90406
1	Aircraft Armaments Inc. ATTN: John Hebert York Rd & Industry Lane Cockeysville, MD 21030
2	BLM Applied Mechanics Consultants ATTN: A. Boresi H. Langhaar 3310 Willett Drive Laramie, WY 82070
1	Martin Marietta Labs ATTN: J.I. Bacile Orlando, FL 32805
1	H.P. White Laboratory 3114 Scarboro Road Street, MD 21154
1	CALSPAN Corporation ATTN: E. Fisher P.O. Box 400 Buffalo, NY 14225
1	FMC Corporation Ordnance Engineering Div San Jose, CA 95114
1	Kaman-TEMPO ATTN: E. Bryant 715 Shamrock Road Bel Air, MD 21014
1	General Electric Company ATTN: D.A. Graham M.J. Bulman Lakeside Avenue Burlington, VT 05402
1	S&D Dynamics, Inc. ATTN: Dr. M. Soifer 755 New York Avenue Huntington, NY 11743
1	Southwest Research Institute ATTN: P.A. Cox 8500 Culebra Road San Antonio, TX 78228

DISTRIBUTION LIST

No. of Copies	Organization
1	Southwest Research Institute ATTN: Mr. T. Jeter 8500 Culebra Road San Antonio, TX 78228
1	University of Dayton Research Institute ATTN: S.J. Bless Dayton, OH 45469
1	University of Delaware Dept of Mechanical Engr ATTN: Prof. H. Kingsbury Newark, DE 19711
3	University of Illinois Aeronautical & Astronautical Engineering Department ATTN: A. Zak (2 cys) D.C. Drucker 101 Transportation Bldg. Urbana, IL 61801
2	University of Iowa College of Engineering ATTN: R. Benedict E.J. Haug Iowa City, IA 52240
1	Rutgers University Department of Mechanical and Aerospace Engineering ATTN: Dr. T.W. Lee P.O. Box 909 Piscataway, NJ 08854
1	University of Wisconsin Mechanical Engineering Dept ATTN: Prof. S.M. Wu Madison, WI 53706
4	Los Alamos Scientific Lab ATTN: Tech Library J. Taylor R. Karpp U. Kocks P.O. Box 808 Albuquerque, NM 87115
4	Sandia National Laboratories ATTN: L. Davison P. Chen L. Bertholf W. Herrmann Albuquerque, NM 87115
4	SRI International ATTN: D. Curran L. Seamman Y. Gupta G. Abrahamson 333 Ravenswood Avenue Menlo Park, CA 94025
1	Drexel Institute of Technology ATTN: P.C. Chou 32nd and Chestnut Streets Philadelphia, PA 19104
1	New Mexico Institute of Mining and Technology Terra Goup Socorro, NM 87801
2	Rensselaer Polytechnic Inst. Dept of Mechanical Engr Troy, NY 12181
2	Southwest Research Institute Dept of Mechanical Sciences ATTN: U. Lindholm W. Baker 8500 Culebra Road San Antonio, TX 78228
7	Dir, USAMSAA ATTN: AMXSY-D AMXSY-MP, H. Cohen AMXSY-G, E. Christman AMXSY-G, R. Conroy AMXSY-OSD, H. Burke AMXSY-LM, J.C.C. Fine APG, Md 21005
1	Dir, USAHEL ATTN: A.H. Eckles, III APG, MD 21005

DISTRIBUTION LIST

No. of Copies	Organization
2	Cdr, USATECOM ATTN: AMSTE-CE AMSTE-TO-F APG, 21005-5055
11	Dir, USACSL, Bldg. E3516, EA ATTN: SMCAR-RSP-A SMCAR-MU SMCAR-SPS-IL SMCAR-CLD SMCAR-MS SMCAR-CLN SMCAR-CLN-D, L. Shaff SMCAR-CLN-D, F. Dagostin SMCAR-CLN-D, C. Hughes SMCAR-CLN, J. McKivrigan SMCAR-CLJ-L APG, MD 21005

DISTRIBUTION LIST

No. of Copies	Organization
12	Administrator Defense Technical Info Center ATTN: DTIC-DDA Cameron Station Alexandria, VA 22304-6145
1	HQDA (SARD-TR) Washington, D.C. 20310
1	Commander U.S. Army Material Command ATTN: AMCDRA-ST 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander U.S. Army Laboratory Command ATTN: AMSLC-DL Adelphi, MD 20783-1145
1	Commander Armament R&D Center U.S. Army AMCCOM ATTN: SMCAR-MSI Picatinny Arsenal, NJ 07806-5000
1	Commander Armament R&D Center U.S. Army AMCCOM ATTN: SMCAR-TDC Picatinny Arsenal, NJ 07806-5000
1	Director Benet Weapons Laboratory Armament R&D Center U.S. Army AMCCOM ATTN: SMCAR-LCB-TL Watervliet, NY 12189
1	Commander U.S. Army Armament, Munitions and Chemical Command ATTN: SMCAR-ESP-L Rock Island, IL 61299
1	Commander U.S. Army Aviation Sys Cmd ATTN: AMSAV-DACL 4300 Goodfellow Blvd. St. Louis, MO 63120
1	Director U.S. Army Aviation Research and Technology Activity Ames Research Center Moffett Field, CA 94035-1099
1	Commander U.S. Army Communications - Electronics Command ATTN: AMSEL-ED Fort Monmouth, NJ 07703
1	Commander U.S. Army Missile Command ATTN: AMSMI-RD Redstone Arsenal, AL 35898-5000
1	Director U.S. Army Missile Command ATTN: AMSMI-AS Redstone Arsenal, AL 35898-5000
1	Commander U.S. Army Tank Automotive Cmd ATTN: AMSTA-TSL Warren, MI 48090
1	Director U.S. Army TRADOC Analysis Cmd ATTN: ATAA-SL White Sands Missile Range, NM 88002
1	Commandant U.S. Army Infantry School ATTN: ATSH-CD-CSO-OR Fort Benning, GA 31905
1	AFWL/SUL Kirtland AFB, NM 87117
1	Air Force Armament Laboratory ATTN: AFATL/DLODL Eglin AFB, FL 32542-5000

DISTRIBUTION LIST

Organization

Aberdeen Proving Ground

Dir, USAMSAA
ATTN: AMXSY-D
AMXSY-MP, H. Cohen
Cdr, USATECOM
ATTN: AMSTE-TO-F
Cdr, CRDC, AMCCOM
ATTN: SMCCR-RS -A
SMCCR-MU
SMCCR-SP-IL

USER EVALUATION SHEET/CHANGE OF ADDRESS

This laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers below will aid us in our efforts.

1. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

2. How, specifically, is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

3. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. _____

4. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) _____

BRL Report Number _____ Division Symbol _____

Check here if desire to be removed from distribution list. ___

Check here for address change. ____

Current address: Organization _____
Address _____

----------FOLD AND TAPE CLOSED----------

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T(NEI)
Aberdeen Proving Ground, MD 21005-5066

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

BUSINESS REPLY LABEL
FIRST CLASS PERMIT NO 12062 WASHINGTON D C
POSTAGE WILL BE PAID BY DEPARTMENT OF THE ARMY

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T(NEI)
Aberdeen Proving Ground, MD 21005-9989